VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřicí a řídicí techniky

Testování implementace sběrnice PROFINET do systémů Simotion

Testing Profinet Communication Network Implementation to Simotion System

2010 **Bc. Martin Dovica**

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

……

Datum odevzdání diplomové práce: 7.5.2010

Poděkování

Touto formou bych chtěl poděkovat všem kolegům ze společnosti ANF DATA spol. s.r.o., se kterými jsem spolupracoval během vytváření této diplomové práce a doc. Ing. Jiřímu Koziorkovi Ph.D. za výpomoc při realizaci této diplomové práce.

Abstrakt

Diplomová práce se zabývá problematikou integračního testu při vývoji komunikačního rozhraní PROFINET implementovaného do systému Simotion. Teoretická část je zaměřena na základní vlastnosti rozhraní PROFINET a kontrolérů řady Simotion. Praktická část rozebírá specifikaci testovaných vlastností, návrh a realizaci testovacích aplikací v konfiguračním prostředí Scout a Step7 a také realizaci automatických testů v programovacím jazyce C# a Python. Systémy Simotion v kombinaci s rozhraním PROFINET slouží pro řízení a synchronizaci víceosých pohybů v průmyslové automatizaci.

Klíčová slova

PROFINET, Simotion, Integrační test, Hard real time systém

Abstract

The present thesis deals with the issue of an integration test in the development of the PROFINET communication interface implemented into the Simotion system. The focus of the theoretical part is on the basic characteristics of the PROFINET communication interface and Simotion series controllers. The practical part looks into the specification of the tested properties, design and realization of testing applications in the Scout and Step7 configuration environment as well as the realization of the automatic tests in the C# and Python programming languages. Simotion systems in combination with PROFINET communication interface serve to control and synchronize mutliaxial movements in industrial automation.

Keywords

PROFINET, Simotion, Integration test, Hard real time system

Seznam použitých symbolů a značek

CBA – Komponentně orientovaná automatizace

COM – Komponentový objektový model

FTP – Komunikační protokol pro přenos souborů

Hard Real Time – Systém s tvrdou časovou odezvou

HMI – Rozhraní komunikace člověk/stroj

IRT – Komunikace v reálném čase se synchronizovanou aplikací

IT – Informační technologie

NRT – Časově nezávislý

PG/PC – Rozhraní programovatelný automat/konfigurační počítač

PLC – Programovatelný logický automat

RT – Komunikace v reálném čase bez synchronizované aplikace

TCP/IP – Protokol kontroly přenosu dat transportní vrstvy/ Internet protokol síťové vrstvy

Obsah

1 Úvod 1

2 Cíle diplomové práce 3

3 Použité nástroje 4

3.1 Síť PROFINET **4**

3.1.1 Charakteristika **4**

3.1.2 Oblast použití **5**

3.2 Řídicí systém Simotion **14**

3.2.1 Charakteristika **14**

3.2.2 Využití PROFINETU v systémech Simotion **15**

4 Specifikace testovaných vlastností a realizace projektů 17

4.1 Definice testů **17**

4.1.1 Požadavky a možnosti **17**

4.1.2 Testované vlastnosti **18**

4.2 Realizace projektů **20**

4.2.1 Projekt MAN **20**

4.2.2 Projekt 2xD435 **25**

4.2.3 Projekt 5xP350 **27**

4.2.4 Projekt P350 + 2xET200 **29**

5 Automatizované testy 32

5.1 Možnosti automatizace **32**

5.1.1 Možnosti k nasazení **32**

5.1.2 Postup návrhu **33**

5.1.3 Rozbor okruhů k zautomatizování **35**

5.2 Realizace **36**

5.2.1 Popis modulů **36**

5.2.2 Skripty automatizovaných testů **41**

6 Závěr 47

1 Úvod

Diplomová práce vznikla na základě zadání společnosti ANF DATA spol. s r.o., a Siemens Copany, která se zabývá především vývojem software pro nové produkty společnosti Siemens. Práce řeší konkrétní potřeby testovací laboratoře při vývoji a implementaci rozhraní PROFINET do systémů Simotion. Především se jedná o testování a zajištění zvýšení kvality ovladačů komunikačních karet implementovaných do kontrolérů řady Simotion.

Kontrolér Simotion společně s rodinou zařízení Sinamics dokáže synchronizovat víceosé soustavy a tento systém pak může nahradit mechanickou převodovku. Rozhraní PROFINET je univerzální deterministická komunikační sběrnice, která v těchto systémech umožňuje zastoupit funkci mechanické převodovky. Výjimečnost těchto systémů spočívá v tom, že se jedná o Hard Real Time systém. Komunikace je zde synchronizována společně s vykonávaným programem ve všech částech distribuovaného řídicího systému s pevně danou periodou cyklu. Většina úkolu zpracována v této diplomové práci řeší problematiku testování synchronizace komunikace vůči aplikaci vykonávanou v kontroléru. Toto testování je realizováno za pomoci již implementovaných mechanismů v kontroléru (elektronická převodovka), ale také za pomoci mnou navržených aplikačních programů.

Vývojová cesta každého produktu v sobě zahrnuje i jeho testování. Diplomová práce v sobě shrnuje část výsledků mé práce v integračním testu. Integrační test je prvním testem, který přichází po zakomponování nových vlastností do produktu. Tento test předává své výsledky zpět do vývoje, kde jsou nalezené chyby opraveny a jsou přidány nové vlastnosti. Tento postup se opakuje tak dlouho, než dojde k odladění všech specifikovaných vlastností. Před předáním produktu koncovému zákazníkovi dochází k hloubkovějšímu testování všech vlastností (systémový test). I pro tuto poslední vlnu testů mohou být využity výsledky diplomové práce.

Na začátku realizace práce jsem stál před požadavkem na testování konkrétních vlastností existujících systémů ve formě prototypů a vývojových verzí ovládacích programů. Nejdříve bylo nutné se s těmito systémy naučit pracovat, poznat principy jejich funkce a naučit se je konfigurovat. Proto se první část práce zabývá teoretickým rozborem komunikačního rozhraní PROFINET. Následuje krátký rozbor vlastností kontrolérů Simotion.

Druhá část diplomové práce se zabývá praktickým řešením zadání. Praktická část spočívá v definování několika testovaných vlastností, v návrhu možnosti jejich realizace a v samotné realizaci v programovém prostředí Scout a Step7 (konfigurační programy pro kontroléry Simotion). Prvním úkolem je navržení testu izochronní komunikace mezi 13 kontroléry D435 za pomoci integrovaných funkcí elektronické převodovky mezi reálnými motory. Tato testovací aplikace simuluje tiskařský stroj firmy MAN Roland, který je v současné době již vybavován prvními prototypy kontroléru Simotion s komunikačním rozhraním PROFINET. Druhým úkolem je navrhnout a realizovat testy izochronní komunikace pro tři různé sestavy za pomocí mnou navržených aplikací v programové prostředí Scout. První sestavou jsou dva kontroléry D435, které spolu komunikují pomocí izochronního komunikačního rozhraní PROFINET v rozmezí komunikačního taktu 1 ms až 4 ms. Druhou sestavou je pět P350 s komunikačním cyklem 250 us až 4 ms. Poslední test realizovaný v programovém prostředí Scout se zabývá komunikací dvou jednotek vzdálených vstupů/výstupů ET200S a jedním kontrolérem P350. Návrh a realizace všech testů jsou popsány

v této práci slovně, pomocí blokových schémat, Petriho sítí a ukázek zdrojového kódu. Nedílnou součástí testování je i fyzické sestavení testovacích soustav.

Třetí část diplomové práce se zabývá automatickými testy kontroléru Simotion v kombinaci s komunikačním rozhraním PROFINET. První dva popisované automatické testy realizují simulaci chybových stavů. Každý z vyvíjených produktů by měl být odolný vůči chybovým stavům, v tomto případě se jedná o výpadek napájení kontroléru a rozpojení komunikačního kabelu. Testování těchto chybových stavů může být provedeno manuálně, ale jen v omezené míře. Požadavkem na odolnost vůči odpadnutí napájení je, že kontrolér musí správně nastartovat a navázat komunikaci i po několika stovkách restartů. V případě rozpojení komunikačního kabelu se jedná o tisíce cyklů rozpojení a spojení. Proto jsem navrhl automatické testy ve skriptovacím jazyce Python a programovacím jazyce C# tak, aby byly schopny v co nejkratším čase realizovat simulaci těchto chybových stavů. Samotná fyzická realizace simulace chybových stavů se provádí pomocí nestartovacích relé a speciálního rozpojovače komunikačního kabelu. Třetí automatický test má na starosti automatickou konfiguraci projektu. Jedná se o automatickou změnu aplikačního cyklu pro projekt realizující testování izochronní komunikace mezi 5 kontroléry P350. Manuální testování změny času v rozmezí 250 us až 4 ms s krokem 125us bylo velice časově náročné. Tento test využívá modifikace exportu projektu, který pak za pomoci interních funkcí Scout a Step7 je nahrán do kontrolérů. Všechny tři testy jsou plně automatické a univerzální tak, aby mohly být použity pro různé testovací soustavy. Kapitola automatických testů je popsána slovně, pomocí diagramů tříd, Petriho sítí a ukázek zdrojového kódu.

2 Cíle diplomové práce

Diplomová práce si klade za cíl přiblížit vlastnosti komunikačního rozhraní PROFINET a systémů Simotion. Tyto poznatky pak dále rozvést a ukázat jak se v praxi konkrétně realizuje kontrola kvality specifikovaných vlastností.

Seznámení se s principem fungování samotné komunikace PROFINET je důležité pro následnou detekci chyb a hledání jejich příčiny. V této práci nejsou principy rozváděny příliš detailně, jen v takové míře, která je potřebná pro pochopení základních principů funkce. Kontroléry rodiny Simotion jsou přiblíženy více z pohledu jejich využitelných vlastností, než z pohledu technologie řešení jejich interních funkcí. Integrační test má na starost hledat chyby především v části kontroléru, který zajišťuje komunikaci přes PROFINET.

Po pochopení základních vlastností použitých prostředků si klade tato práce za cíl přiblížit postup práce při návrhu testů deterministické komunikace mezi kontroléry Simotion P350, D435 a jednotkou vzdálených vstupů / výstupů ET200S. Kontrola deterministické komunikace může být zajištěna buď pomocí již implementovaných nástrojů nebo díky vlastním aplikačním programům. Všechny projekty, které zajišťují základní kontrolu deterministické komunikace jsou realizovány v konfiguračním prostředí Step7 a Scout. V diplomové práci jsou popsány celkem 4 projekty.

Dalším cílem práce je zhodnocení možností realizace automatických testů pro simulaci chybových stavů a jejich následná realizace. Tyto automatické testy se skládají z více komponent. Některé z nich jsou realizovány pomocí programovacího jazyka C#, ze kterých je vygenerována dll knihovna s COM rozhraním. COM rozhraní pro některé z elementárních funkcí automatizovaných testů jsou vybrány z toho důvodu, aby mohly být využity v kterémkoli z jiných programovacích jazyků. Hlavní část automatizovaných testů je pak realizována ve skriptovacím jazyku Python. Tento jazyk pro hlavní testy je vybrán z historického důvodu. Všechny popsané součásti automatizovaných testů přiblíží vývojové diagramy a ukázky zdrojových kódů. Nejsložitější automatizovaný test je popsán jako poslední. Nejedná se o testování chybových stavů, ale o automatickou konfiguraci projektu v prostředí Scout. Tento test využívá interních funkcí prostředí a díky nim je schopen automaticky konfigurovat komunikační cyklus v daném projektu. Po úpravě projektu tento skript provede nahrání do kontroléru a zpustí aplikaci umístěnou v kontroléru pro testování deterministické komunikace.

Posledním, ne méně důležitým, cílem je zhodnocení dosažených výsledků. Všechny projekty jsou realizovány tak, aby byly snadno rozšiřitelné a pochopitelné jiným uživatelem. Proto poslední část rozvádí možnosti vylepšení již hotové práce.

3 Použité nástroje

3.1 Síť PROFINET

3.1.1 Charakteristika

Jedním z aktuálních požadavků moderní automatizace je zajištění homogenní komunikace na všech úrovních struktury řízení výroby, tzn. od úrovně akčních členů a přístrojové techniky v technologickém provozu, přes úroveň řízení (programovatelných automatů) a výše až do nadřazených stupňů zpracování a vyhodnocování technologických dat. Standardizované propojení, jednotná správa sítě, osvědčené techniky z oblasti IT a komplexní diagnostika mohou znamenat úsporu ve fázích projektování i při oživování zařízení či v běžném provozu. Výhody, které poskytují průmyslové sběrnicové systémy (fieldbuses), a na druhé straně standardizované funkce ze světa IT založené na Ethernetu, by měly být využívány současně pomocí jednotného komunikačního systému. Proto byl mezinárodní organizací Profibus International definován rozsáhlý standard PROFINET, vycházející ze standardu Ethernet, který provozním jednotkám umístěným přímo v technologickém provozu otevírá zcela nové možnosti. Společnost Siemens má s komunikačním standardem PROFINET rozsáhlé zkušenosti. Jeho základní vlastnosti jsou shrnuty v následujícím textu.

PROFINET je odpověď na nové požadavky a trendy:

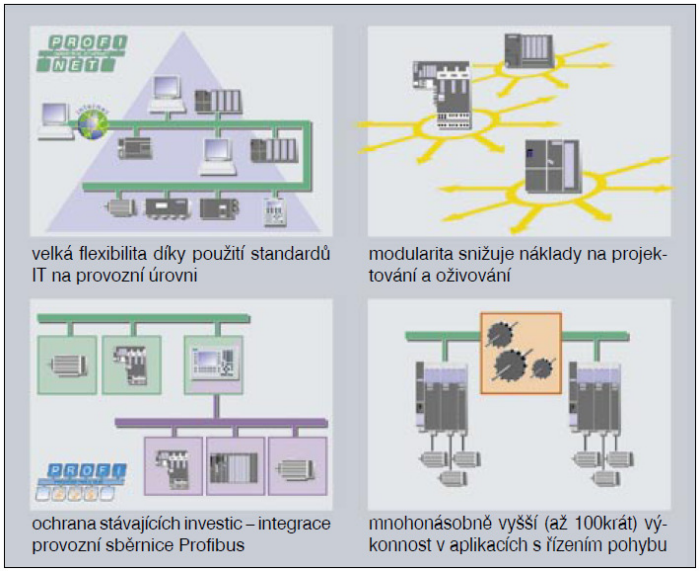
- je to otevřený standard definovaný mezinárodní organizací PROFIBUS International
- je založen na průmyslovém Ethernetu
- využívá TCP/IP a zavedené standardy IT
- má jednoduchou administraci a diagnostiku
- umožňuje komunikovat v reálném čase
- je možná integrace existujících průmyslových sběrnic (PROFIBUS, Interbus)

Oproti jiným řešením nabízí především:

- připojení více uzlů
- několikanásobně větší výkonnost v aplikacích s řízením pohybu
- jednotný sběrnicový systém pro všechny úrovně podnikové automatizace na bázi sítí Ethernet
- přístup k technologickým datům pomocí standardních kancelářských nástrojů
- větší komfort dálkového přístupu k technologickým zařízením a diagnostiky na dálku
- bezdrátovou komunikaci prostřednictvím průmyslových sítí WLAN

PROFINET je tedy standardizovaný, otevřený, na výrobci či dodavateli nezávislý komunikační systém pro všechny úrovně průmyslové automatizace, založený na průmyslovém Ethernetu. Je schopen zajistit odezvu v reálném čase a umožňuje i izochronní přenosy dat, např. pro řízení pohybu. Do sítě PROFINET je možné bez problémů začlenit distribuovanou přístrojovou techniku různých výrobců – projektování a tvorba distribuovaných automatizačních struktur jsou velmi efektivní a nezávislé na konkrétním dodavateli přístrojů. Instalaci sítě lze navrhnout s ohledem na průmyslové prostředí. Administrace systému je jednoduchá a k diagnostice se využívají běžně známé služby IT. Pamatováno je i na zabezpečení proti neautorizovanému přístupu a manipulaci s daty. K dispozici je i komunikace odolná

proti selhání (failsafe). Z uvedených důvodů je PROFINET vhodným a perspektivním řešením pro všechny aplikace a průmyslová odvětví spojená s automatizací diskrétních i kontinuálních výrobních procesů.



Obr.1 Základní charakteristika sběrnice PROFINET - převzato z [1]

3.1.2 Oblast použití

PROFINET umožňuje integrovat dosavadní průmyslové sběrnicové systémy, jako např. PROFIBUS, bez nutnosti modifikace a změny existujících zařízení. Tím je zajištěna ochrana současných investic zákazníka. V komunikačním systému PROFINET může zákazník použít pro všechny úrovně struktury řízení pouze jedinou komunikační sběrnici. To přispívá k úspoře nákladů na skladování, instalaci, údržbu i na vyškolení personálu.

Používání strojů či jejich částí jakožto inteligentních technologických modulů v koncepci automatizace založené na komponentech (CBA – Component Based Automation) znamená úsporu při vývoji a oživování technologických zařízení. Koncepce CBA přímo předpokládá využití komunikačního systému PROFINET, a díky tomu je distribuovaná automatizace v tomto pojetí ekonomicky velmi zajímavá.

Vedle zmíněné úspory nákladů dává jednotná komunikační struktura na bázi Ethernetu navíc široký prostor pro technické inovace. Například parametrizaci, diagnostiku či vizualizaci lze realizovat prostřednictvím webových služeb. Díky dálkovému přístupu přes Ethernet lze nabídnout nový způsob poskytování servisu a celosvětové technické podpory a zajistit tak vyšší provozní spolehlivost technologických zařízení.

Pokračující další vývoj tohoto komunikačního systému dává uživatelům dlouhodobou perspektivu. Certifikáty vydávané organizací Profibus International garantují plnou kompatibilitu produktů. Sortiment komponent pro PROFINET nabízených společností Siemens zahrnuje řídicí systémy, distribuované vstupně/výstupní a systémy HMI s patřičnými rozhraními a stejně tak i síťové komponenty, jako jsou průmyslové přepínače (switch), průmyslové prvky pro bezdrátové (rádiové) sítě WLAN, průmyslové bezpečnostní prvky pro ochranu podnikových

sítí a velký počet pasivních síťových komponent a prvků pro rychlou a snadnou montáž kabelových rozvodů.

Pro vývoj aplikací se používá standardní a osvědčený software z produkce Siemens s názvem Step 7. Pro distribuované automatizační systémy podle koncepce CBA je k dispozici nový softwarový nástroj Simatic iMap – na výrobci nezávislý editor pro spojování komponent (technologických modulů).

Internetové prostředí a web

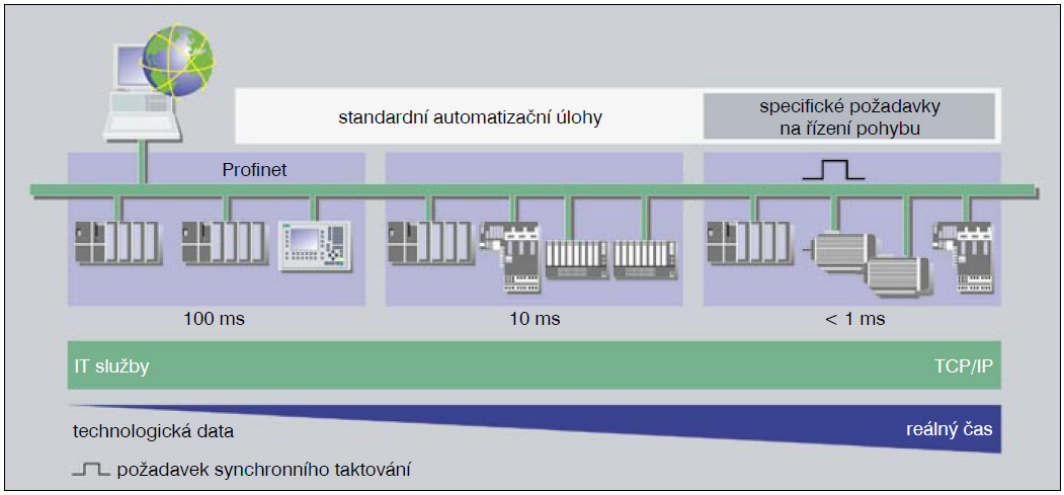
PROFINET je přístupný i pro webové klienty. Přístup je založen na standardních jazycích a protokolech používaných v síti Internet, jako jsou HTTP (Hypertext Transfer Protocol), XML (Extensible Markup Language), HTML (Hypertext Markup Language) nebo skriptování. Data jsou přenášena ve standardizovaném tvaru (HTML, XML) a zobrazována s použitím standardizovaných funkcí, aplikací či prohlížečů (jako např. Netscape, Microsoft Internet Explorer, Opera atp.). Tím lze pohodlně začlenit informace ze zařízení připojených do komunikačního systému PROFINET do moderních multimediálních informačních systémů. Pro jednotky systému PROFINET jsou tudíž využitelné všechny výhody webové integrace z oblasti IT, jako např.:

- používání webových prohlížečů jako jednotných grafických operátorských rozhraní
- přístup k informacím bez omezení počtu klientů a nezávislý na umístění informací
- klienty nezávislé na platformě
- snížení nákladů na instalaci a aktualizaci klientů

Komunikace prostřednictvím systému PROFINET

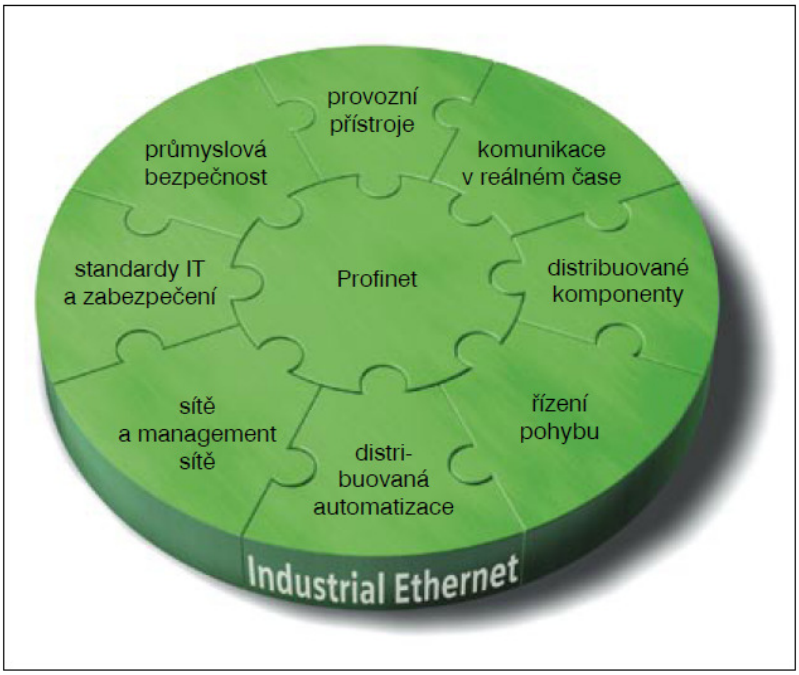
PROFINET je součástí normy IEC 61158 (Provozní sběrnice pro použití v systémech průmyslového řízení) a vychází z mezinárodního standardu Ethernet (IEEE 802.3). Jedná se tedy o „přepínaný“ Fast Ethernet (100 Mb/s). PROFINET je založen na standardech informační techniky, jako např. na protokolu TCP/IP, který je z principu asynchronní. Pro účely provozní automatizace ovšem PROFINET poskytuje také možnost komunikace v reálném čase s typickou odezvou 1 až 4 ms; to odpovídá požadavkům většiny běžných úloh, které se v automatizaci vyskytují. Těchto parametrů, v podstatě srovnatelných s dosavadními typy průmyslových sběrnic, je dosahováno pomocí optimalizace přenosu dat a řízení priorit. Pro komunikaci v reálném čase lze rovněž používat standardní síťové komponenty.

Další rozšíření rozsahu aplikací představuje izochronní přenos, který je určen pro specifické aplikace vyžadující rychlou odezvu a především přesné taktování – typickým příkladem je řízení pohybu (obr. 2). Izochronní přenos dat, označovaný jako IRT (isochronou real-time), je schopen dosáhnout časového cyklu sběrnice 250 μs s časovou nejistotou (jitter) menší než 1 μs; to je podstatná vlastnost např. pro současné řízení většího počtu pohybových os. Izochronní datový přenos je implementován pomocí čipů ASIC ERTEC (Application Specific Integrated Circuit – Enhanced Real-Time Ethernet Controller). Tyto integrované obvody jsou součástí příslušných koncových zařízení či síťových komponent.



Obr.2 PROFINET – komunikace - převzato z [1]

Díky tomu, že v jednom komunikačním systému lze realizovat asynchronní, synchronní i izochronní přenosy (TCP/IP, RT a IRT), lze zajistit přenos jak časově kritických dat, tak i dat časově nekritických, což umožňuje vždy optimální přizpůsobení daným požadavkům při současném zachování homogenity a jednotného komunikačního standardu v celém podniku či výrobním provozu. To je hlavní předností této koncepce komunikačního systému. Kromě toho je PROFINET skutečně komplexní komunikační systém, který splňuje veškeré požadavky pro moderní průmyslovou automatizaci (obr. 3).



Obr.3 PROFINET – komplexní standart pro průmyslový Ethernet v automatizaci - převzato z [1]

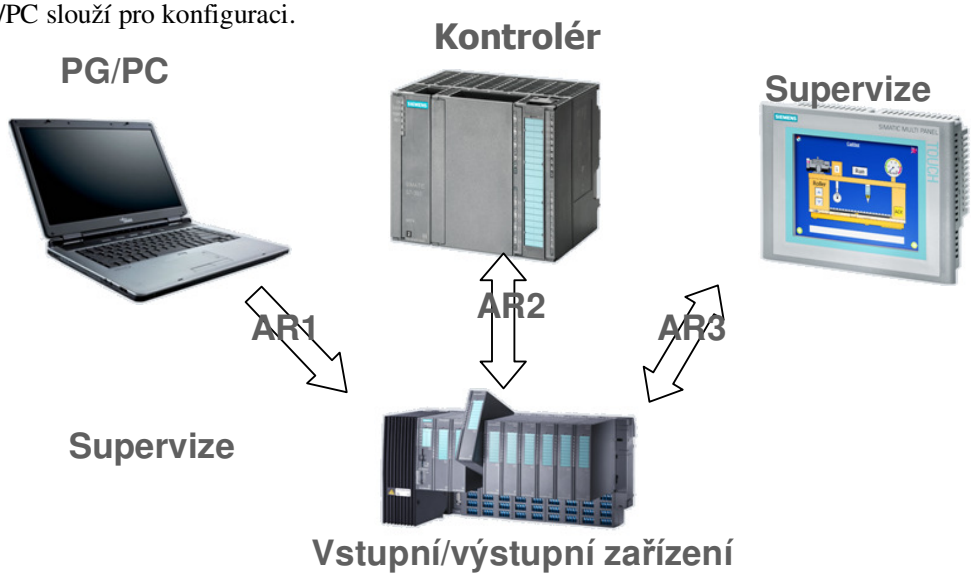
PROFINET IO

PROFINET IO (označováno také PROFINET RT) je komunikační sběrnice, která akceptuje distribuované aplikace a není zde kritická otázka časové precizní determiničnosti (pro precizní RT komunikaci se používá PROFINET IRT, dále v textu). Může plně nahradit automatizační aplikace na bázi stávající sběrnice PROFIBUS, pouze sběrnicové rozhraní musí být nahrazeno. Z hlediska použití a konfigurace systému se však jedná o velice podobnou komunikační sběrnici jako je PROFIBUS.

Nejčastější místa využití PROFINET IO

- ovládání relé
- kontrola ventilu
- pohony
- poziční enkodéry
- zobrazovací jednotky
- měření a analýza zařízení
- roboti
- bezpečnostní inženýrství (světelné bariéry, bezpečnostní tlačítka atd.)

Základní komunikační vazby jsou znázorněny na obr. 4 . Real-time komunikace (RT) je znázorněna jako AR2 a představuje datovou výměnu mezi stanicí IO Controller a stanicí/stanicemi IO Device. Do RT komunikace spadá i posílání a potvrzování alarmů. AR2 znázorňuje také Non-real-time (NRT) komunikaci, kam spadá parametrizace, konfigurace a acyklické služby. Stanice IO Supervisor představuje inženýrskou stanici PC, HMI apod., PG/PC slouží pro konfiguraci.

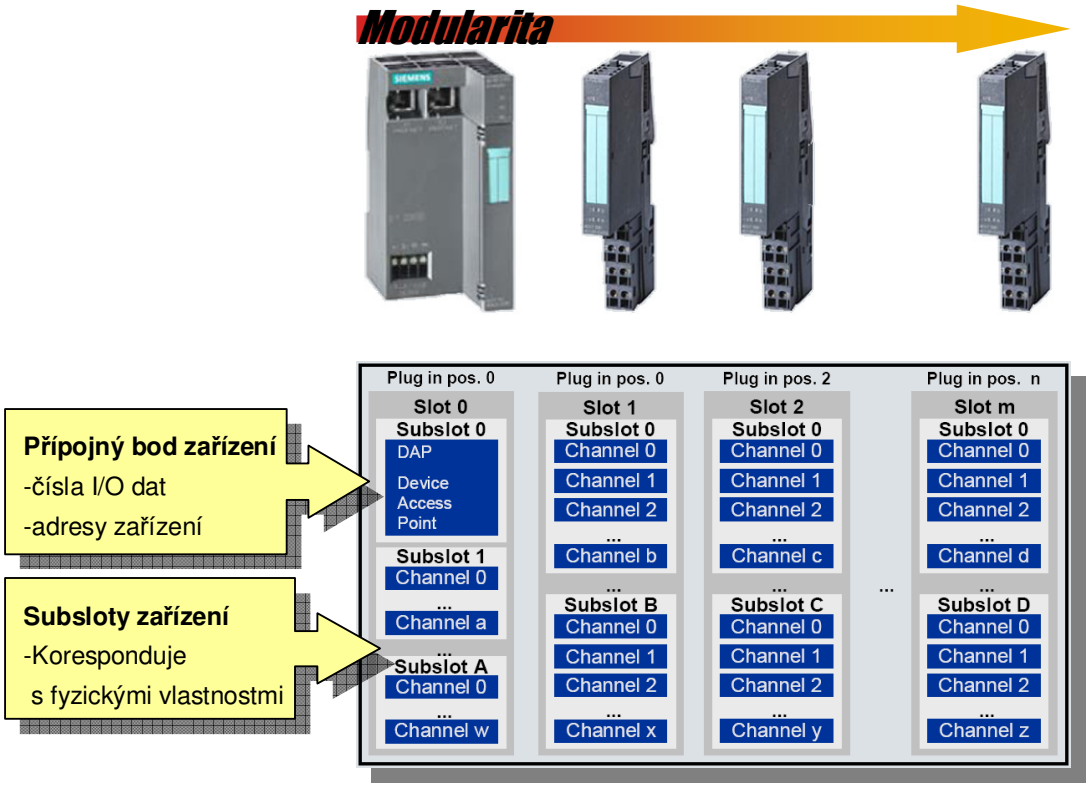


Obr. 4 Základní aplikační relace (AR) PROFINET IO

AR1 - Kontrolér AR	IO data, konfigurační data a Alarmová data
AR2 - Supervize AR	Čtení IO dat, zápis jen pro nastavování
AR3 - Implicitní AR	Pouze čtení dat

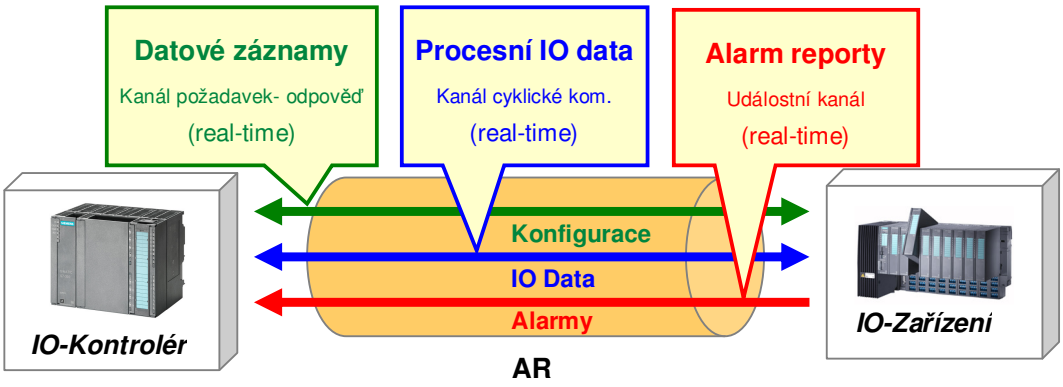
Data jsou v rámci PROFINET IO rozdělena do objektů I/O dat (IO Data Objects) a objektů záznamů (Record Data Objects). Objekty IO dat (IO data) se přenášejí cyklicky bez potvrzování s využitím principu provider/consumer v rámci RT komunikace. Každý objekt IO dat je přiřazen určitému modulu/submodulu v rámci zařazení. Všechna data ze zařízení jsou přenášena v jednom Ethernetovém rámci a každému objektu IO dat je také přiřazen jeho status. Tato stavová informace se vyhodnocuje na straně příjemce, který tak může určit, zda jsou data platná. Objekty záznamů představují diagnostickou informaci, popis zařízení (funkce identi_cation & maintenance), záznamy alarmů a chyb apod. Obr. 5 ukazuje, jak jsou data rozdělena do slotů/subslotů, což odpovídá fyzickému členění zařízení do modulů/submodulů. Znázorněné zařízení obsahuje komunikační modul, který zajišťuje připojení k síti PROFINET IO a je v něm implementovaný komunikační protokol. Je vždy umístěn ve slotu O. IO data pro cyklickou komunikaci se získávají z datových modulů, které jsou umístěny v dalších slotech.

Příkladem může být analogový vstupní modul se čtyřmi kanály, umístěný ve slotu 1. Data z jednotlivých kanálů mohou být předávána v rámci jednoho submodulu, tj. všechny analogové hodnoty vystupují jako jeden datový objekt, nebo se může přenášet každá analogová hodnota jako samostatný datový objekt v rámci samostatného submodulu. To jaká možnost bude využita, je určeno aplikací v IO Controlleru neboli v Hardwarové konfiguraci PLC.

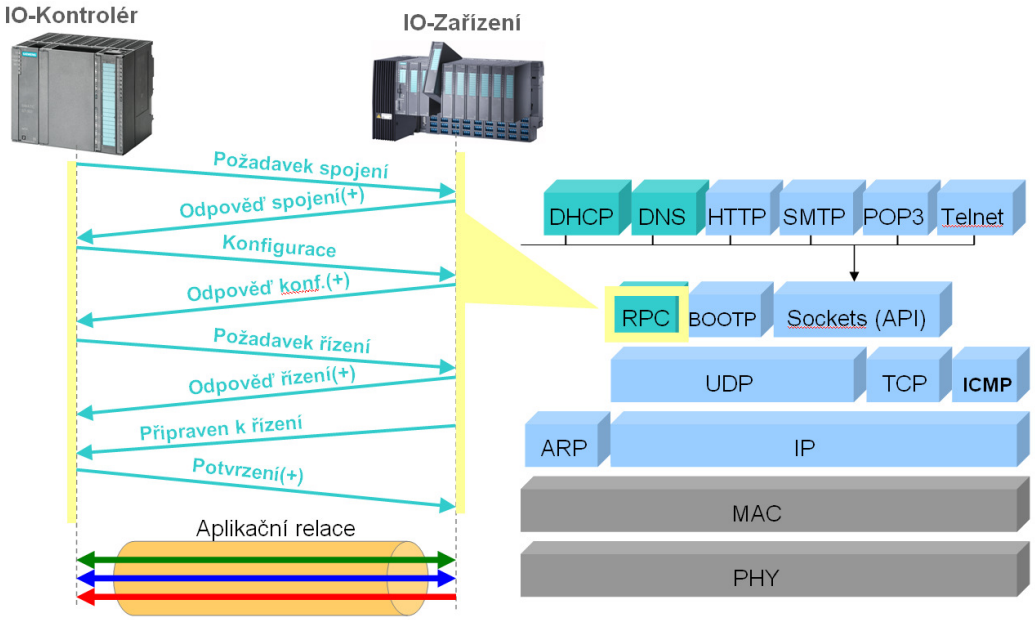


Obr. 5 Model řízení IO device - převzato z [3]

Každá komunikace mezi zařízeními IO Controller a IO Device existuje v rámci aplikační relace (AR), jak ukazuje obr. 6. Po založení aplikační relace jsou založeny komunikační relace (IO CR) a předány parametry zařízení pomocí relace datových záznamů (Record data CR). Relace vzniknou na základě požadavku na spojení (Connect.req) viz obr. 7. Po odeslání pozitivního potvrzení na Connect.req začne datová výměna, i když jsou data zatím označena jako neplatná. Musí totiž následovat ještě fáze parametrizace zařízení pomocí komunikace Write. Fáze parametrizace je zakončena zprávou DControl.req od zařízení IO Controller, která je potvrzena zprávou DControl.res, čímž je fáze navazání spojení dokončena. Datová výměna s platnými daty může začít po komunikaci CControl.req/CControl.res. Detailní průběh komunikace je vidět na obr. 7. Cyklická datová výměna je zde znázorněna modrou čárou, NRT parametrizační komunikace zelenou a alarmy červenou čárou.



Obr. 6 Aplikační komunikační relace PROFINET IO

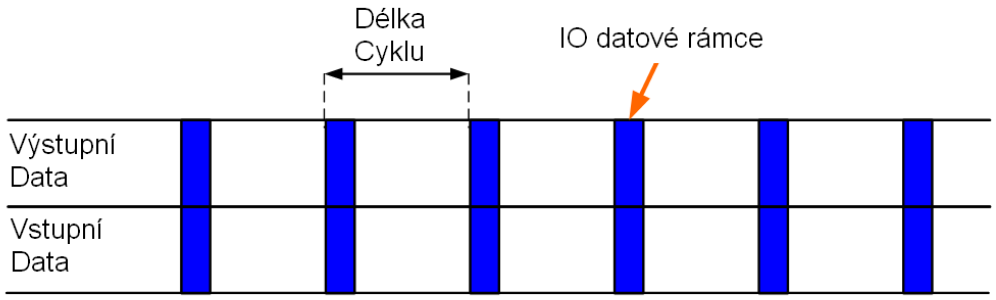


Obr. 7 Průběh navázaní komunikace – RPC protokol a jeho postavení vůči ostatním protokolům a vrstvám PROFINET - převzato z [3]

PROFINET IO IRT

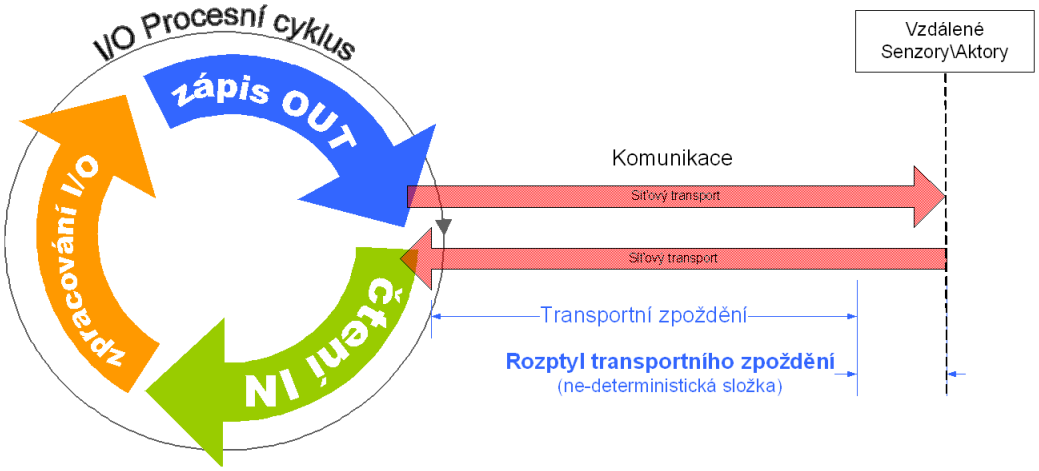
Tato komunikace by mohla být srovnávána se sítí metra ve městě. K dosažení cílové stanice ze stanice počáteční je k dispozici mnoho tras. Mezi nimi je jedna nebo více přestupních stanic a je možno určit jednu trasu, kterou bude cíle dosaženo nerychleji. Takto se dá také chápat komunikace IRT v síti. Komunikační cesta se proto plánuje v průběhu návrhu. Komunikace v „červeném intervalu“ (přenos cyklických dat) je založena na plánování konfigurace předem, tj. kromě nastavení informace o koncových uzlech určených pro výměnu dat, tak i o středových uzlech určených pouze pro přenos informace. Komunikační balíky jsou předávány výhradně na základě plánovacího algoritmu definovaného v IEC 61158. Pokud systém vyžaduje RT_CLASS_3 komunikaci (jiné označení pro IRT) musí být cyklus sběrnice rozdělen na část „červený interval“ a část „zelený interval“ (UDP /IP) v průběhu návrhu. Toto rozdělení probíhá automaticky v některém z konfiguračních nástrojů (např. Step7). Jestliže se topologie systému změní, je třeba znova naplánovat celou komunikaci.

PROFINET IO IRT nabízí oproti PROFINET IO (PROFINET RT) precizní časově deterministickou (izochronní) komunikaci mezi kontrolérem a device (případně komunikaci kontrolér – kontrolér). Je zde synchronizována jak komunikace (vysílání a příjem) tak i vykonávání jednotlivých programů v rozsahu 250 us až 4 ms. Důležitá je fixní topologie pro využití PROFINET IRT, podle nastavené topologie se plánuje časování posílání dat. Je možno posílat až 1440 uživatelských bajtů v jednom cyklu.



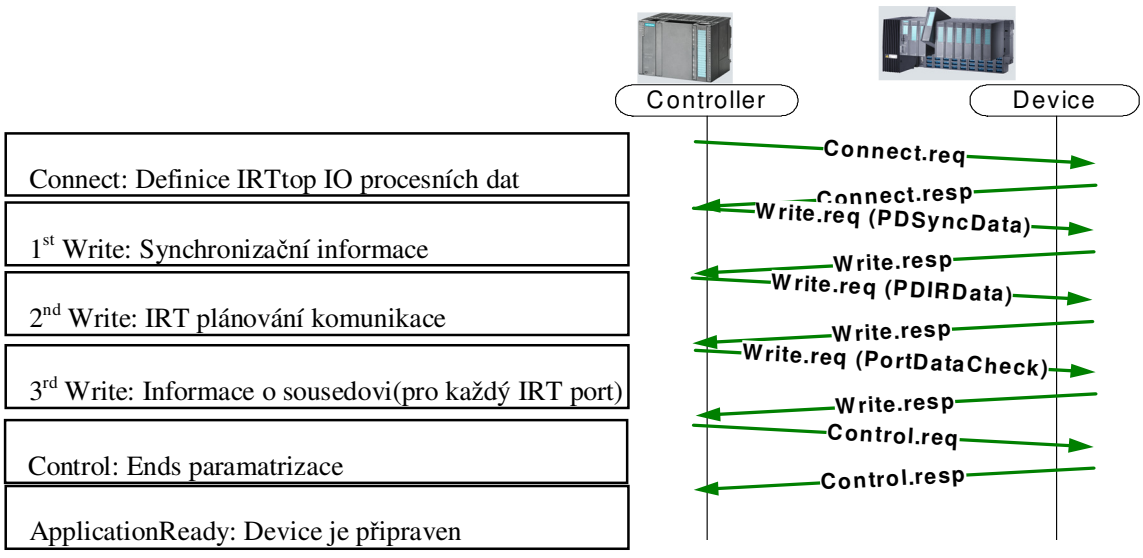
Obr. 8 Znázornění synchronizace komunikačního rámce

Cílem PROFINET IO IRT je, mimo jiné, kompenzovat dopravní zpoždění na komunikační sběrnici. Filozofie takovéto kompenzace je, že předpověditelné zpoždění, v tomto případě dopravní zpoždění, může být kompenzováno. Na komunikační cestě dvou uzlů způsobuje zpoždění samotná rychlost šíření se elektrického signálu, ale také ostatní síťové prvky umístěné do topologie (např. switch). Celkové dopravní zpoždění a výpočet plánování posílání dat se opírá jak o neprojektovanou topologii, včetně předpokládané délky kabelů, tak i o měření zpoždění na začátku komunikace.

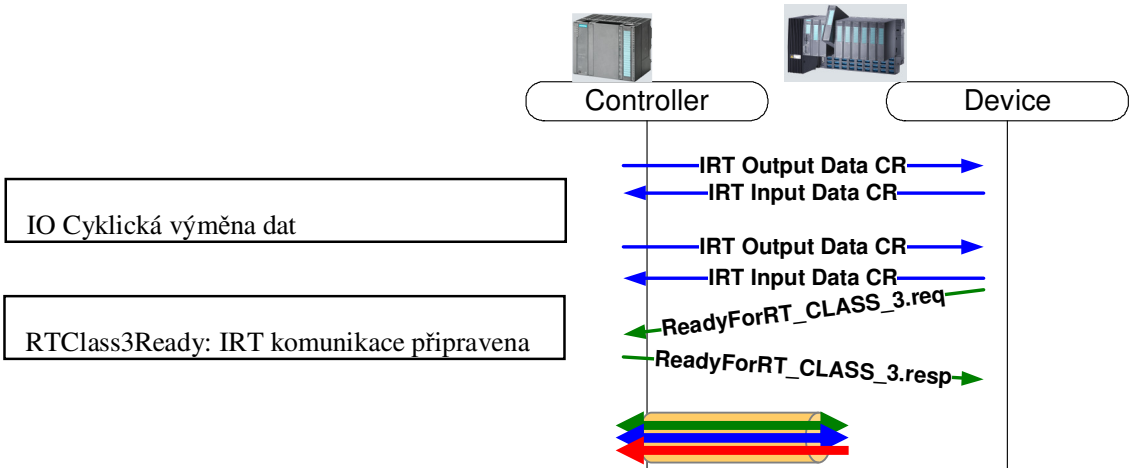


Obr. 9 Procesní cyklus komunikace a zpracování dat - převzato z [2]

Navázaní komunikace u PROFINET IO IRT je obdobné jako u RT verze. Navázaní aplikační relace mezi kontrolérem a device je zobrazeno na následujících dvou obrázcích. Nejprve dochází k navázaní spojení, to znamená, že dojde k přiřazení IP adresy a Device Name na základě definované topologie. Poté dojde k výměně synchronizačních informací, informací ohledně IRT plánování a zápisu dat ohledně komunikačních sousedů. Po přijetí ApplicationReady od zařízení může dojít k výměně cyklických dat. V topologii musí být vždy jeden synchronizační vedoucí (master - většinou jeden kontrolér) a ostatní zařízení mají roli synchronizačního podřízeného (slave) [pro tuto kapitolu byly čerpány informace z použité literatury – 1,2,3,4].



Obr. 10 Navázání aplikační relace u PROFINET IRT - převzato z [2]



Obr. 11 Navázání aplikační relace u PROFINET IRT - převzato z [2]

Po navázaní aplikační relace se v tomto kanálu přenášejí (na obrázku 11 znázorňuje aplikační relaci hnědožlutý válec).

- Data Record CR
  - Konfigurační datový kanál
- IO Data CR
  - IO cyklická data
- Alarm CR
  - Alarmy (např. subslot nebo slot neexistuje – byl odpojen)
- Sync Services
  - Obsluha synchronizace
- Network management services
  - Síťové služby

3.2 Řídicí systém Simotion

3.2.1 Charakteristika

Řídicí systém SIMOTION je z hlediska programování založen na principu multitaskingového systému. Na pozadí běží cyklicky zpracovávaný blok stejně jako hlavní cyklický blok OB1 v PLC SIMATIC. V něm lze například zpracovávat logiku a sekvence ne přímo závislé na pohonech a polohování. Další úlohy (tasky) jsou startovány, pozastavovány, znovu spouštěny a ukončovány nezávisle na tomto cyklu a právě ony jsou určeny pro řešení jednotlivých pohybů. Tasky se programují v grafických blocích podobných vývojovému diagramu nebo v programovacím jazyku podobném např. Pascalu, případně v grafickém programovacím prostředku FBD. Programování systému probíhá ve vývojovém prostředí SCOUT, v němž je integrováno rozhraní pro pohony (Drive ES) a STARTER pro práci s měniči SINAMICS. Nadstavbou může být např. konfigurační software pro programování vaček CamTool.

Z hlediska hardwarové platformy řídicí systém SIMOTION existuje ve třech provedeních označených jako C, P a D. Toto označení pochází z názvů platforem Controller, PC, Drive. Jednou vytvořený program pro ŘS SIMOTION je díky propojení komponent sběrnicovým způsobem nezávislý na platformě. Jednotlivé platformy mají své specifické přednosti, všechny však splňují základní myšlenku sjednocení třech základních funkcí: logického řízení, řízení typu Motion Control a vykonávání technologických funkcí.

Řídicí systém SIMOTION spojuje v sobě tyto funkce:

- funkce PLC
- Motion Control (synchronizace, polohování otáčivých os)
- technologické funkce (regulace teploty, tlaku)

Řídicí systém SIMOTION C

Je stavební forma podobná řídicím systémům SIMATIC S7-300, vhodná pro aplikace s více centrálními vstupy/výstupy, na něž se použijí běžné karty od systému SIMATIC S7-300. Isochronní sběrnicí PROFIBUS DP, případně PROFINET se připojí frekvenční měnič, nadřazená řídicí úroveň, operátorský panel apod. Procesor SIMOTION C má označení C230. Typicky může kontrolovat 8 os.

Řídicí systém SIMOTION D

Je stavební součástí víceosého frekvenčního měniče SINAMICS S120 a z tohoto hlediska nahrazuje jeho řídicí jednotku. Kombinovaný modul se tedy skládá z vlastního jádra SIMOTION a z druhého procesoru, který přebírá funkci řídicí jednotky frekvenčního měniče SINAMICS S120. Pro komunikaci s okolím je možno využít integrované rozhraní PROFIBUS nebo rozšiřující kartu CBE30, která v sobě integruje 4 porty pro PROFINET. Vstupy a výstupy, které mají přímou vazbu k řízeným osám měniče SINAMICS S120 jsou připojeny pomocí vlastní sběrnice měniče typu Drive CLiQ. Typicky může kontrolovat 8 os.

Řídicí systém SIMOTION P

Je průmyslové panelové PC s rozhraním na frekvenční měnič přes isochronní PROFIBUS DP nebo PNIO_MC_BOARD (rozšiřující karta umožňuje použití 4 portů sběrnice

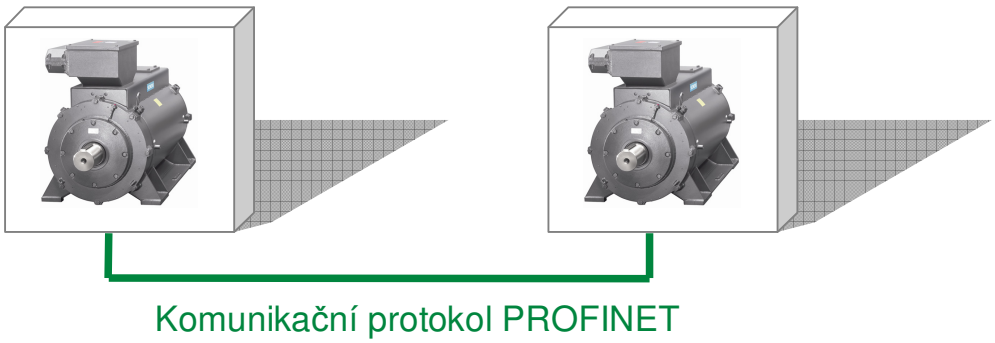
PROFINET, využívá interní PCI sběrnici). Výhodou je, že na výkonném PC mohou kromě aplikace SIMOTION běžet i další programy s vysokými nároky na výpočetní výkon. Standartně se jedná o operační systém Windows XP, vedle kterého běží Simotion runtime. Vstupy a výstupy jsou realizovány výlučně jako necentrální, prostřednictvím periferních jednotek SIMATIC ET 200S. Provedení panelového PC SIMOTION P má označení P350. Typicky může kontrolovat 16 os.

V diplomové práci jsou požity řídicí systémy typu D a P, výhradně se sběrnicí PROFINET IRT.

Obr. 12 Ilustrační vzhled jednotlivých typu Řídicích systému Simotion, zleva C, D, P - převzato z [5]

3.2.2 Využití PROFINETU v systémech SIMOTION

Přínosem sběrnice PROFINET do řídicích systému SIMOTION je především její časově deterministická odezva. Celý systém může tak tvoři Hard Real Time systém, jehož podstatou je to, že všechny pracovní cykly musí být zpracovány přesně v daném čase (ani později, ani dříve). Testování spolehlivosti přenosu dat v přesně daných intervalech je dominantní součástí navrhovaných testů v této diplomové práci (více v následujících kapitolách). Jak již bylo řečeno v předchozí kapitole, komunikace PROFINET ITR zajišťuje synchronizaci komunikace i vykonávaných aplikací. To v praxi znamená, že synchronizaci vykonávaných programů lze využít k synchronizaci otáčivých os, PROFINET slouží jako ekvivalent mechanické převodovky. Výhodou je tak souběh několika os na velkou vzdálenost, převodový poměr, vačka se mohou měnit dynamicky v průběhu vykonávaní programu. Toto řešení nabízí snažší údržbu bez nutnosti seřizování vůlí převodů [pro tuto kapitolu byly čerpány informace z použité literatury – 5].



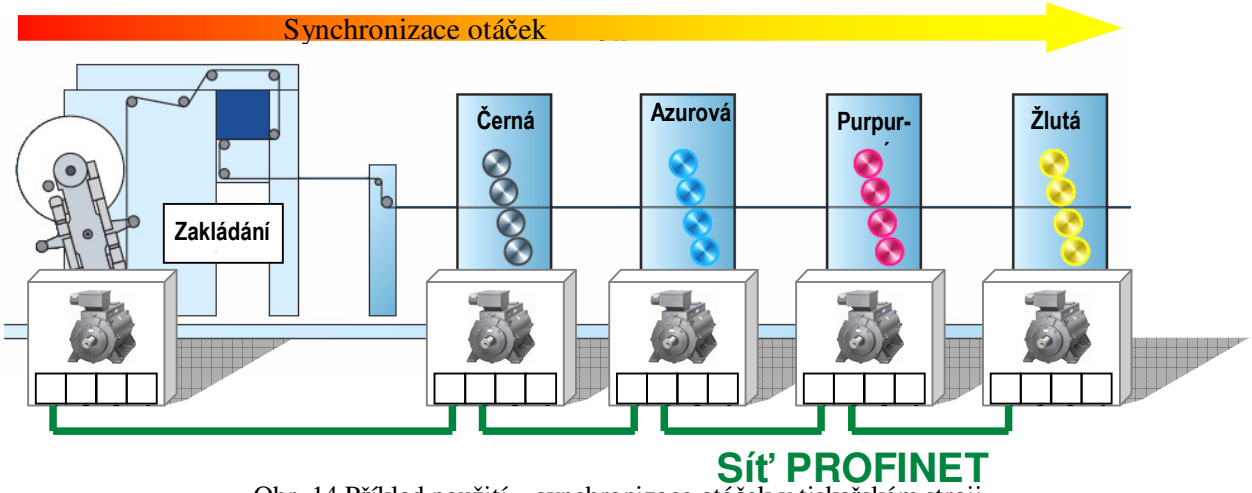
Obr. 13 Elektronický vázaný souběh - převzato z [2]

Souběh os může probíhat v těchto režimech:

- základní
  (polohování dle aktuálních hodnot, otáčková vazba, váčkový spínač, měřicí vstup)
- poziční kontrola
  (polohová vazba, programovatelný otáčivý pohyb, traverzní profil, pohyb na pevnou zarážku, regulace síly/tlaku, referencování, korekce a superimponované pohyby)
- převodovka
  (funkce master osy - využito v této diplomové práci, úhlová synchronizace, elektronický převod, absolutní/relativní, synchronizace/de-synchronizace – spojka, suporimponování a offsety, distribuované synchronní operace
- CAM
  (elektronická vačka, tabulková, nebo polynomická - 6-tého řádu, pohyby dle VDI 2143 - německá norma pro dynamické vačky, absolutní/relativní, cyklická/necyklická, svaliny, offset, superimponování, přepínaní

Typy Os:

- elektrické osy (servo, vektor) s digitální, nebo analogovou žádostí
- hydraulické osy
- osy s krokovými motory



Obr. 14 Příklad použití – synchronizace otáček v tiskařském stroji
- převzato z [2]

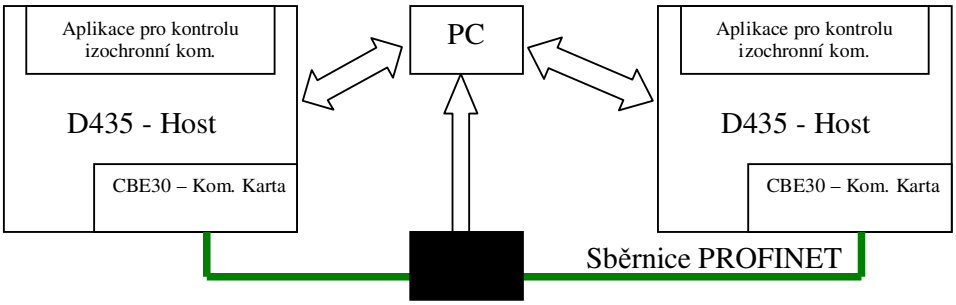
4 Specifikace testovaných vlastností a realizace projektů

4.1 Definice testů

4.1.1 Požadavky a možnosti

Hlavním požadavkem na testování sběrnice PROFINET implementované do řídicích systému Simotion je zajištění izochronní komunikace, stability a spolehlivosti. S ohledem na tyto aspekty jsou navrženy následující testy. Tato diplomová práce se nezabývá základní funkcionalitou a podmínkami nutnými pro nastavení a nakonfigurování komunikace pomocí sběrnice PROFINET, jako je například přiřazení IP adres, jmen zařízení, správné funkce konfiguračních programů, možnosti nastartování izochronní komunikace. Splnění jednotlivých podmínek je součástí integračního testu, ale v následujícím textu již předpokládejme, že tyto aspekty jsou splněny a plně fungují.

Navrhovaní testů vychází ze specifikací vlastností komunikační sběrnice a konkrétního řídicího systému. Například jednou z vlastností může být maximální počet bajtů, které si můžou dva PLC vyměnit s daným taktem. V tomto testu je pak zahrnuto testování izochronnosti sběrnice na nejnižší úrovni. Jedná se především o driver pro komunikační kartu CBE 30 nebo PCIPN_Board a její návaznost na „Host“, kterým je řídicí systém Simotion. Test pak vypovídá o vlastnostech celého systému jako takového. V případě nalezení chyby (např. rozpadla se komunikační synchronizace po 1 hodině) je třeba zajistit reprodukci této chyby a pomocí analyzačních nástrojů dohledat, kde a v jaké části systému mohla tato chyba nastat. Pod pojmem analyzační nástroj je možno si představit různé druhy programů, které dokáží zachytávat komunikaci na sběrnici, zapisovat dění uvnitř PLC (Logovat). Následující obrázek ilustračně přibližuje jak vypadá testovaný systém.



Obr. 15 Základní schéma pro testování izochronní komunikace

Jak z předchozího obrázku vyplývá, je důležité zajistit kontrolu správné funkce komunikace a chování celého systému na úrovni uživatelské aplikace.

To je možno dvěma způsoby:

- využití již vyvinutých nástrojů kontrolujících izochronnost a stabilitu, tzn. napojit jednotlivé PLC na reálné motory, nakonfigurovaní os a jejich synchronizací(osa je objekt v programovém prostředí Scout a umožňuje synchronizaci otáček více motorů)
- vytvoření vlastní aplikace, která zajišťuje komunikaci mezi dvěma nebo více uzly a kontroluje správnou výměnu dat

Oba uvedené způsoby testování komunikace jsou předmětem této diplomová práce a jsou popsány v následujícím textu.

4.1.2 Testované vlastnosti

Jak je uvedeno v předchozím textu, testované vlastnosti vycházejí ze specifikace normy PROFINETU a vlastností jednotlivých řídicích systémů. Vybrané vlastnosti pro testování:

1. synchronizace 16 os s využitím funkce gearing (převodovka)
2. maximální počet přenesených dat mezi dvěma Simotion řady D435 (požadavek je 2540 bajtů oběma směry za 1 ms)
3. komunikace mezi pěti Simotion P350 s taktovacím cyklem v rozmezí 250 us až 4 ms
4. isochronní komunikace mezi Simotion P350 a jednotkou vzdálených vstupů a výstupů ET200 s cyklem 250 us
5. spolehlivost náběhu synchronizace po restartu
6. spolehlivost náběhu synchronizace komunikace po rozpojení komunikačního kabelu

První testovaná vlastnost vychází z již realizovaného projektu tiskařských strojů firmy Man Roland. Jedná se o velkou rotační tiskárnu, kde jsou jednotlivé osy synchronizované pomocí komunikační sběrnice. V minulosti byla komunikace jednotlivých řídicích systémů zajištěna předchůdcem sběrnice PROFINET a to sběrnicí PROFIBUS. Firma Man Roland v dnešní době začíná nové tiskařské stroje osazovat komunikací na bázi PROFINETU, a proto je nutné v integračním testu vývoje ovladačů pro kartu CBE30 zapracovat test, který simuluje reálný tiskařský stroj této firmy. Více bude uvedeno dále v kapitole realizace testů – projekt MAN.

Druhá testovaná vlastnost vychází ze specifikovaných vlastností řídicího systému a návaznosti na přenos dat. Jak je napsáno v předchozím textu, sběrnice PROFINET dokáže přenášet až 1440 bajtů oběma směry při taktu 250 us. Řídicí systém Simotion D435 dokáže pouze komunikovat s taktem 1 ms a je méně výkonným sourozencem D445, který už zvládá takt 500 us. Omezení maximálního přenosu dat 2540 bajtů v jednom cyklu pro rodinu Simotion D4xx vychází z předpokladu, že při 500 us PROFINET IRT zvládá přenést 2880 bajtů. Proto byla zvolená maximální přenosová zátěž o něco nižší, než je hranice samotného komunikačního média. Teoreticky při vyšších taktech cyklu by mohla mít přenášená data vetší objem, ale z mně neznámých důvodů nelze pro mezikontrolérovou komunikaci nakonfigurovat více, než 10 bloků po 254 bajtech. Tímto testem se bude zabývat projekt 2xD435.

Nejvýkonnějším PLC z řady Simotion nese označení P350. Jedná se o průmyslové PC na bázi operačního systému Windows XP. Do tohoto operačního systému je doinstalovaný

program SimaticNet, který je již hostitelským programem pro samotný program realizující PLC. Do PCI slotu PC je zapojená komunikační karta PNIOMC_Board zajišťující hardwarovou podporu samotného PROFINETU. Úskalím tohoto řešení je opět reakce na izochronní aplikaci a komunikaci. Minimálním časem cyklu tohoto PLC je 250us. Projekt 5xP350 se zabývá aplikací zajišťující dlouhodobé testy stability komunikace. V kapitole 5. je rozebrán program pro automatickou konfiguraci taktů komunikace (250 us až 4 ms s krokem 125 us).

Projekt P350 **+** 2xET200 se zabývá testováním komunikace mezi řídicím systémem a jednotkou vzdálených vstupů a výstupů.

Poslední dva body ze specifikace testovaných vlastností otevírají otázku automatizovaných testů pro restarty kontrolérů a rozpojování komunikačních kabelů. Tyto testovaná vlastnosti vycházejí z předpokladu, že kdykoliv v průmyslu může dojít k výpadku napájení kterékoliv součásti komplexního řídicího systému, případně k rozpojení kabelu. Každá součást takového systému musí po obnovení napájení (zapojení komunikačního kabelu) naběhnout ve stanoveném čase a být připravena k zpracovávání aplikačního programu. Z hlediska těchto testu není ani tak zajímavé jak řešit problém výpadku jednotlivé součásti z pohledu aplikace, nýbrž z pohledu samotného náběhu systému do stavu „připraven pro zpracovávaní programu“. Jelikož ve specifikovaných vlastnostech je, že systém Simotion musí zajistit izochronní přenos dat i po několika tisících restartech, případně rozpojení kabelů, došlo k navržení a realizaci automatických testů. Více v kapitole: 5. Automatické testy.

4.2 Realizace

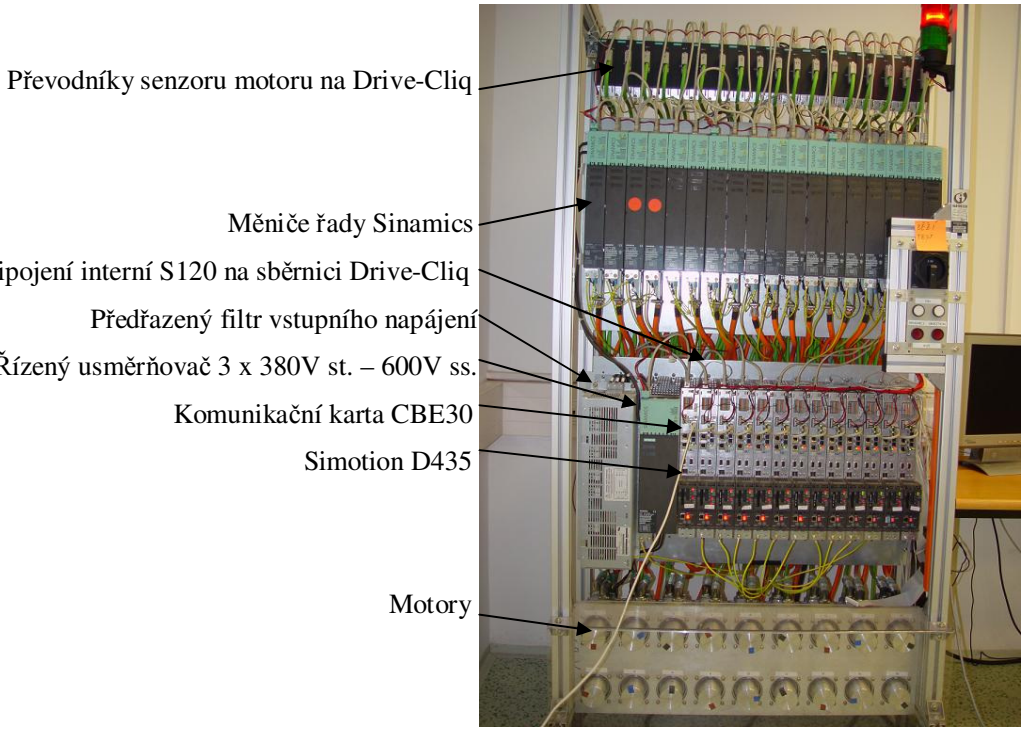
4.2.1 Projekt MAN

První popisovou realizací je projekt nazvaný MAN. Jak již bylo zmíněno v předchozí kapitole, jedná se o konfiguraci, která testuje simulaci tiskařského stroje od firmy MAN Roland.

V tomto projektu je použito:

- 13 x Simotion D435 verze C (integrovaný Sinamics S120)
  ON: 6AU1-435-0AA00-0A4
- 13 x CBE30 verze E (komunikační karta pro PROFINET)
- 18 x Single Motor Modul 9A (motorový měnič)
  ON: 6SL3120-2TE13-0AA1
- 18 x Převodník pro Drive CLIQ – X520
- 1 x Active Line Modul 27A, 16kW (usměrňovač ovládaný pomocí Drive Cliq)
  ON: 6SL3130-7TE21-6AA0
- 1 x Line Filter 30A, 16kW (filter napájení) ON: 6SL3130-7TE21-6AA0

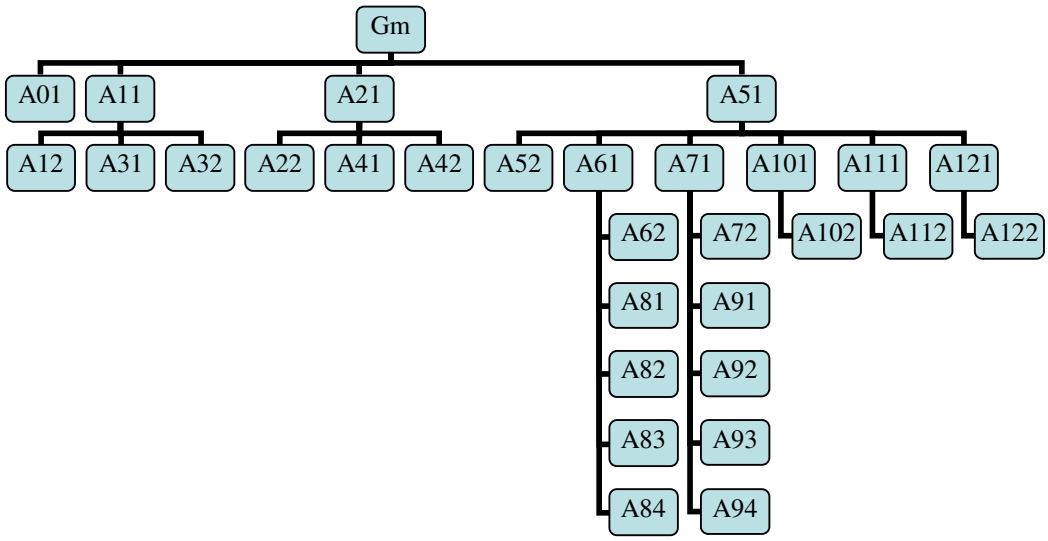
Prvním úkolem bylo sestavit takový rack, který obsahuje navržený hardware. O větší část fyzické realizace se postarala specializovaná firma, která je součástí koncernu Siemens. Doladění počtu použitých kontrolérů, jejich správných verzí a jejich napojení na spínací relé je provedeno mnou (napájecí relé využívá automatizovaný test).



Obr. 16 Fotografie simulace tiskařského stroje

Principem tohoto projektu je synchronizace os na jednu master osu (hlavní osa podle které ostatní synchronizují své otáčení). Master osa je ovládána první D435, která je současně synchronizační master z hlediska komunikace (taktuje přenos v celé nakonfigurované doméně). Topologie propojení PROFINETU v tomto projektu je navržená liniová, tohoto faktu pak využívá rozložení synchronizací os tak, aby jednotlivými uzly „teklo“ co nejvíce dat. Z důvodu rozložení výkonu nemůže být všech 18 motorů synchronizováno na jednu osu přímo, zpravidla jedna D435 při cyklu 1 ms zvládá synchronizovat pouze 4 až 5 os (v závislosti na použité synchronizaci). Proto je zde navržena struktura synchronizací tak, aby se byl výkon rovnoměrně rozvržen mezi všechna PLC. Toto řešení přináší jednu nevýhodu, a to, že komunikace sice probíhá synchronně, ale regulace závislých os je vždy posunuta o jeden cyklus se vnořením hlouběji do navržené struktury převodovky. Tento fakt je možno v tomto případě zanedbat, jelikož se jedná pouze o simulaci stroje.

Jelikož jedna D435 dokáže synchronizovat až 5 os (v případě reálných os) a fyzicky bylo dostupných pouze 18 motorů, jsou v projektu nakonfigurovány také virtuální osy. Tyto osy nejsou tak náročné na výkon řídicího systému, ale z hlediska komunikace převodovky přes PROFINET jsou stejné jako osy reálné. Virtuální osy se v praxi používají v moment, kdy je třeba některý z motoru odstranit, ale není žádoucí zasahovat do struktury navrženého projektu (virtuální osa muže být mezičlánek „elektronické převodovky“).



Obr. 17 Navržená struktura synchronizací na synchronizační Master osu - GM

Z předchozího obrázku je patrné, že se všechny osy synchronizují na hlavní osu GM. Označení synchronních os Axy koresponduje s označením jednotlivých PLC a číslem osy na tomto PLC. Osy s označením vyšším než A82 jsou virtuální.

Nastavení a aktivování synchronizace otáčení je nastaveno v poměru 1:1. Tento poměr může být měněn dynamicky v běhu programu a to pro každou osu zvlášť. Jestliže je tato převodovka (Gearing) realizována mezi dvěma kontroléry (neděje se tak pouze v rámci jednoho kontroléru), musí docházet k výměně 24 bajtů oběma pro každou převodovku. Těchto 24 bajtů

obsahuje informace o aktuální poloze (směrem k Master ose), o požadované poloze (k synchronizované ose) a další informace umožňující samotnou funkci převodovky. Obsahuje také i informace kontrolující izochronnost těchto dat, tzv. komunikační Watchdog. Tento Watchdog je realizovaný pomocí dvou bajtů, ve které se cyklicky inkrementují (vychází ze specifikace PROFIDRIVE – specifikace telegramů pro komunikaci se servy Sinamics). Každý z komunikační dvojice inkrementuje svůj Watchdog, každý kontroluje svého komunikačního partnera, zda v aktuálním cyklu příjme data s inkrementovaným Watchdog. Jestliže kterýkoliv z komunikačních článků zjistí, že nastala chyba v komunikaci, dojde tak k vyhlášení alarmu o rozpadu komunikační synchronizace a v tomto případě se synchronizovaná osa zastaví. Proto lze pomocí os a jejich převodovek kontrolovat izochronnost komunikace sběrnice PROFINET. Jinými slovy, jestliže se roztočí všechny synchronizované osy a dojde k zastavení jedné z nich, došlo k rozpadu komunikační synchronizace, byť jen v jednom cyklu.

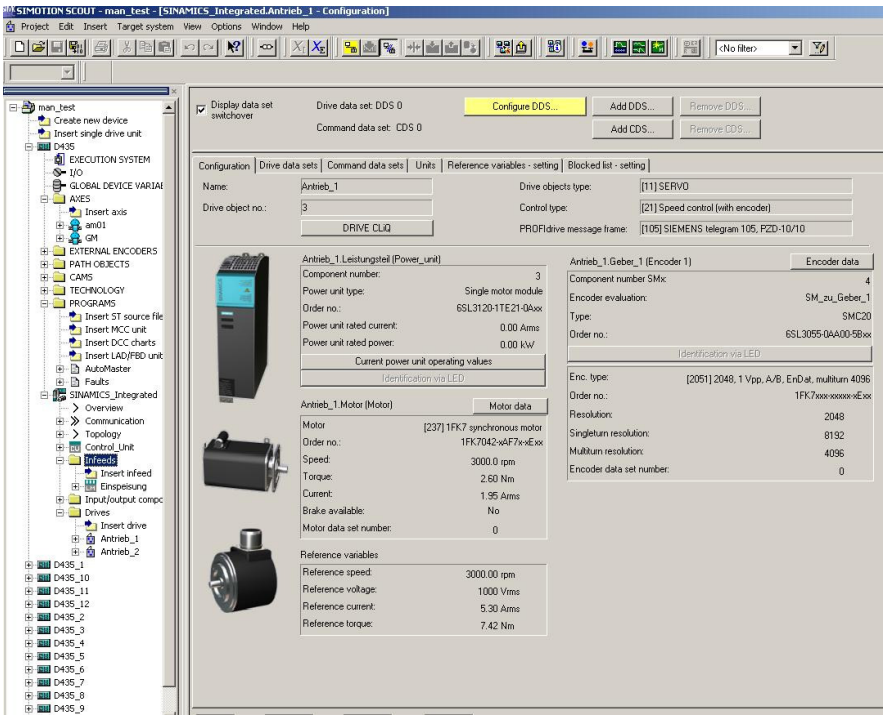
Z hlediska statické stability komunikace dochází ke dvěma typům testů:

- zátěžové komunikační testy
- dlouhodobé testy točení os

Zátěžový komunikační test probíhá tak, že se rozběhne komunikace mezi jednotlivými kontroléry a do kterékoliv CBE30 se připojí externí komunikační zátěž. Může se jednat třeba o cyklické posílání pingů ve více vláknech, FTP zátěž, případně sofistikovaný paket vysílaný pomocí speciálního programu nebo přístroje, který dokáže vysílat komunikační zátěž. CBE30 se musí zachovat tak, že za žádnou cenu nemůže dojít k rozpadu synchronizace PROFINETU i při 100% externí zátěži. Řadič profičet se musí pokusit přenést maximální počet dat v NRT části komunikace (část komunikačního cyklu určená pro necyklická data) a zbytek dat zahodí. Tento test většinou probíhá několik hodin a využívá se Ethernet analyzéru od firmy Fluke, který dokáže generovat jakýkoli standardní paket, případně i uživatelsky definovaný. Nejčastěji jsou využívány ARP pakety.

Jak je popsáno v předchozím textu, pomocí točení os a elektronické převodovky, lze zajistit dlouhodobé testování správné synchronizace sběrnice PROFINET. Obrázek 17 ukazuje navržené schéma hierarchie převodovky jednotlivých motorů na hlavní motor. Pro účel testovaní bylo nutné vytvořit testovací projekt pro všechny části řídicího systému MAN rack. Diplomová práce se nezabývá detaily ohledně konfigurace celého projektu v programovém prostředí Scout a Step7, spíše principem jak tento test funguje. Následující odstavec jen zevrubně přibližuje, jak vypadá konfigurace takového projektu.

Step7 je balík programů umožňující konfiguraci řídicích systému na bázi PLC od firmy Siemens. Program Scout je doplňkovou nadstavbou k Step7. Každý projekt na bázi Simotion se začíná tím, že je potřeba vytvořit hardwarovou konfiguraci (přidat do projektu jednotlivé PLC, nastavit topologii, druh komunikace, IP adresy atd. v programu HW Config – součást Step7), poté je třeba nakonfigurovat jednotlivé použité motory, snímače, komunikační protokoly, osy a synchronizaci mezi osami pro jednotlivé D435 (v programu Scout). Následující obrázek zobrazuje náhled do projektu MAN. V levé části jsou vidět jednotlivé D435. Konkrétně první D435 má nastavené dvě osy (am01, GM), obsahuje dva programy (AutoMaster a Faults – posáno v následujícím textu) a integrovaný Sinamics má nastavený jeden aktivní usměrňovač a dva motor moduly, které mají nastavené dva motory, dva snímače a dva převodníky.



Obr. 18 Náhled na projekt MAN v programu Scout

Po nakonfigurování celého projektu a nahrání do jednotlivých D435 sice naběhne cyklická komunikace a je možno nastavit všechny D435 do stavu RUN, ale stále zde chybí aplikace, která dokáže nastartovat usměrňovač, aktivovat jednotlivé převodovky a roztočit hlavní osu. Pro zajímavost, v případě aktivovaných převodovek se všechny osy snaží synchronizovat svou polohu podle hlavní osy, aniž by se otáčela. Což znamená, že se dá otáčet hlavní osou například rukou a všechny synchronizované osy se natočí také v nastaveném převodovém poměru.

Pro tento projekt je zvolen programovací jazyk ST pro konfiguraci aplikace, svou syntaxí připomíná Visual Basic nebo Pascal. Následující ukázka zdrojového kódu programu přiblíží aplikaci pro první D435. Aktivuje usměrňovač, pošle informaci ostatním D435 o tom, že měnič je připraven (pomocí digitálních I/O – jednotlivé D435 mají „prodrátovány“ své digitální vstupy/výstupy), aktivuje převodovku pro osu am01 a roztočí hlavní osu.

```
//Kontrola cyklické komunikace pro osu GM a am01.
REPEAT
      counter:=counter +1;
      UNTIL (
         GM.sensormonitoring.cyclicinterface=ACTIVE
         AND
         GM.actormonitoring.cyclicinterface=ACTIVE
         AND
         am01.sensormonitoring.cyclicinterface=ACTIVE
         AND
         am01.actormonitoring.cyclicinterface=ACTIVE)
      END_REPEAT;

//Sekvence pro aktivaci usměrňovače.
infeed_control :=16#ff8e;
```

```
REPEAT
      ;
      UNTIL infeed_status = 16#0211 OR infeed_status = 16#0291
      END_REPEAT;
//Samotná aktivace funkce usměrňování.
infeed_control :=16#ff8f;
REPEAT
      ;
      UNTIL infeed_status = 16#1A17
       END_REPEAT;
//Aktivace osy am01
      _MccRetDINT := _enableAxis(axis:=am01, enableMode:=ALL, servoControlMode:=ACTIVE,
servoCommandToActualMode:=INACTIVE, nextCommand:=WHEN_COMMAND_DONE, commandId:=_getCommandId(),
forcecontrolMode:=INACTIVE);

//Aktivace osy GM
      _MccRetDINT := _enableAxis(axis:=GM, enableMode:=ALL, servoControlMode:=ACTIVE,
servoCommandToActualMode:=INACTIVE, nextCommand:=WHEN_COMMAND_DONE, commandId:=_getCommandId(),
forcecontrolMode:=INACTIVE);

//Nastavení nulové polohy pro osu am01.
      _MccRetDINT := _homing(axis:=am01, homingMode:=DIRECT_HOMING, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_MOTION_DONE, commandId:=_getCommandId());

//Nastavení nulové polohy pro osu GM.
      _MccRetDINT := _homing(axis:=GM, homingMode:=DIRECT_HOMING, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_MOTION_DONE, commandId:=_getCommandId());

//Informace pro zbylé D435 o aktivaci měniče a připravenosti hlavní osy pomocí digitálního výstupu.
m_axis_en := TRUE;

//Začátek otáčení hlavní osy dychlostí 600°/s.
      _MccRetDINT := _move(axis:=GM, velocityType:=DIRECT, velocity:= 600,
moveTimeOutType:=WITHOUT_TIME_LIMIT, velocityProfile:=TRAPEZOIDAL, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_ACCELERATION_DONE, commandId:=_getCommandId(),
movingMode:=POSITION_CONTROLLED);

//Aktivace převodovky pro osu am01.
      _MccRetDINT := _enableGearing(followingObject:=am01_GLEICHLAUF, direction:=BY_VALUE,
gearingType:=ABSOLUTE, gearingMode:=GEARING_WITH_FRACTION, gearingRatioType:=DIRECT, gearingNumerator:=1,
gearingDenominator:=1, synchronizingMode:=IMMEDIATELY,
syncProfileReference:=RELATE_SYNC_PROFILE_TO_LEADING_VALUE, syncLengthType:=DIRECT, syncLength:=100,
synchronizingDirection:=USER_DEFAULT, mergeMode:=IMMEDIATELY, nextCommand:=WHEN_AXIS_SYNCHRONIZED,
commandId:=_getCommandId());
```

Výpis zdrojového kódu programu pro projekt MAN

Na ostatních D435 (v projektu D435_1 až D435_12) je umístěn program AutoSlave. Tato aplikace je velice podobná té hlavní AutoMaster, s tím rozdílem, že zde se nespouští usměrňovač po kontrole cyklické komunikace pro dané osy, ale čeká se na povolení zapnutí převodovky, a to změnou stavu z logické nuly na logickou jedničku na nastaveném digitálním vstupu.

Takto vytvořený projekt je možno nahrát do jednotlivých kontrolérů a po postupném přepnutí do stavu RUN se roztočí všechny motory (za předpokladu, že spouštěcí aplikace byly umístěné do systému plánování vykonávání programu). Po roztočení jednotlivých motoru začíná test dlouhodobé spolehlivosti izochronní komunikace. Jestliže se jeden z motorů zastaví, je jasné, že nastal výpadek v komunikaci a je třeba analyzovat tento problém.

Další testovanou vlastností, která se aplikuje v tiskařských strojích Man Roland je tzv. Redundat Sync Master (redundantní synchronizační master) z hlediska synchronizace PROFINETU. V praxi to znamená, že je možno vytvořit projekt tak, že jeden kontrolér je

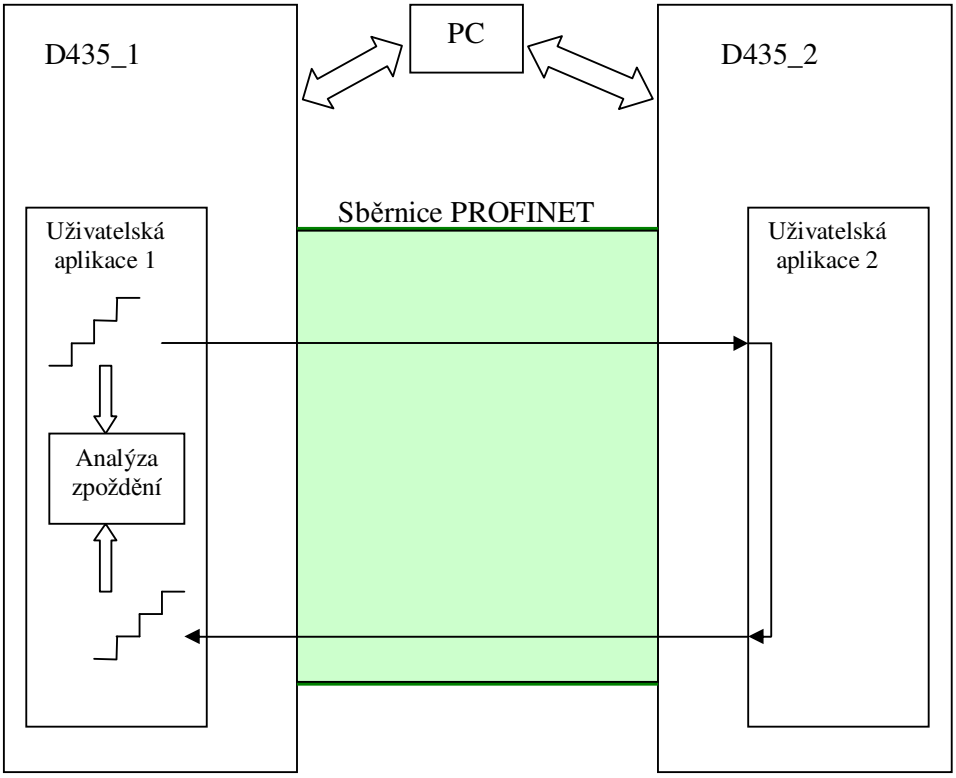
synchronizační master celé komunikace a taktuje celou komunikaci. V případě, že primární master vypadne z komunikace (např. vypnutí, přerušení komunikačního spojení), tak jeho úkol dokáže zastoupit redundantní synchronizační master. Z hlediska projektování to vypadá tak, že se v hardwarové konfiguraci nastaví jeden z kontrolérů jako synchronizační master a jiný jako redundantní synchronizační master.

V mém projektu se jedná o podobnou hardwarovou konfiguraci, první D435 je nastavená jako IRT Sync Master, ale poslední jako IRT Sync Master Redundant. Z hlediska testu synchronizační redundance má každý kontrolér dvě osy. První polovina motorů je spojená převodovkou s prvním kontrolérem, druhá polovina s tím posledním. V případě přerušení komunikace na rozhraní těchto dvou převodovek se musí všechny motory dál otáčet a redundantní synchronizační master komunikace ihned přebírá funkci první D435 z hlediska synchronizace komunikace.

Tato funkce je opět testována pro každou novou verzi firmware karty CBE30 a vychází ze specifikace PROFINETU.

4.2.2 Projekt 2xD435

Projekt nazvaný 2xD435 zajišťuje testování izochronní komunikace mezi dvěma kontroléry Simotion pro maximální počet nekonfigurovatelných bajtů, a to je 2540 oběma směry. Kontrola takového množství dat není možná pomocí os a synchronizace jejich otáčení, protože jedna synchronizace dvou os mezi dvěma kontroléry používá pouze 25 bajtu. Pro využití všech maximálně možných konfigurovatelných dat je třeba využít více než 100 převodovek, toto ale není možné z hlediska výkonu kontroléru. Proto jsem zvolil cestu vlastní aplikace, která by byla schopná pokrýt kontrolu co největšího počtu dat. Pro tento projekt se využívají dva kontroléry z MAN projektu. Blokové schéma navrženého programu je na následujícím obrázku.



Obr. 19 Blokové schéma aplikace pro testování komunikace mezi dvěma Simotion D435

Principem testování komunikace mezi dvěma D435 je obdobný mechanizmus jako je Watchdog pro převodovku mezi osami. Jedná se o inkrementování posílané hodnoty a analyzování komunikačního a aplikačního zpoždění.

V tomto projektu je nastaveno 2540 bajtů pro komunikaci oběma směry mezi dvěma D435. Jeden z kontrolérů v sobě obsahuje program, který ihned po přechodu do stavu RUN začne inkrementovat hodnotu v rozsahu 0 až 100, ve 100 bajtech rovnoměrně vybraných z nastavené komunikace. Tato data posílá svému komunikačnímu partnerovi, který přijímaná data posílá zpět do prvního Simotion, ten pak analyzuje celkové zpoždění této komunikace. Rozdíl mezi hodnotou dat, která se vrátila do prvního kontroléru a těmi, co se posílají ve stejném časovém okamžiku musí, být právě 2. Jak je popsáno v úvodu této práce, výhoda PROFINETU IRT je právě v jeho izochronnosti a schopnosti synchronizovat vykonávanou aplikaci s komunikací. To znamená, že když se v jednom cyklu aplikace zapíše do výstupu, který je vstupem jinému systému, tak celková synchronizace zajistí, že na vstupu druhého systému jsou aktuální data hned v příštím cyklu. Musí být ovšem zajištěno vykonávaní aplikačního programu v tzv. Servo Task. Umístění aplikace do Servo Tasku umožňuje zpracování programu synchronně s komunikací. Čas, který je určen pro aplikaci v Servo Tasku, je omezen a je pouze zlomkem cyklu komunikace. V řídicích systémech Simotion se nedá dohledat, kolik času je potřeba pro vykonání konkrétní uživatelské aplikace. Ale se snižujícím

se taktem sběrnice musí být aplikace méně náročná, jinak dojde k výpadku ze stavu RUN do STOP z důvodu přetížení kontroléru a nedodržení pravidel pro RT systém. V této aplikaci se proto kontroluje komunikace pouze pro 100 bajtů rovnoměrně vybraných z 2540 bajtů, které jsou celkově posílány. Následující ukázka zdrojového kódu přiblíží řešení kontroly komunikace.

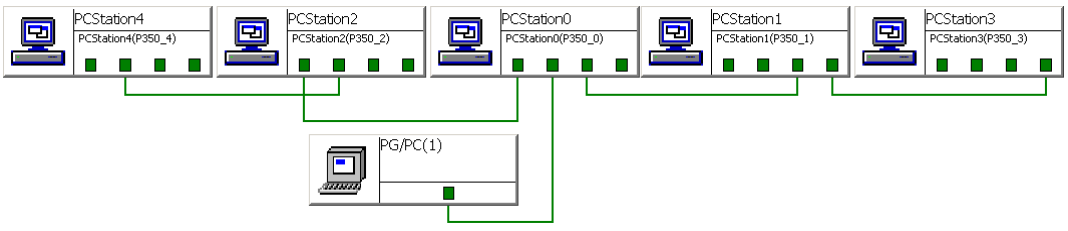
```
//Cyklus for pro zápis do vybraného výstupu a následný výpočet rozdílu mezi aktuálním výstupem a komunikačním
vstupem
FOR i:=0 TO 9 DO
      d435_out_1[i]:=j;
      distance[i]:=(j - (d435_in_1[i]));
       //Ošetření výpočtu rozdílu hodnoty mezi přijímany daty a posílanými po přechodu z hodnoty 100 na 0
      IF(j = 0)
            THEN
            distance[i]:=101 - d435_in_1[i];
      END_IF;
      IF(j = 10)
            THEN
            distance[i]:=102 - d435_in_1[i];
      END_IF;
END_FOR;
```

Výpis zdrojového kódu programu pro projekt 2xD435

Celý program je umístěný v Servo Tasku, který zajišťuje cyklické volání vykonávání zdrojového kódu synchronně s komunikací. Ukázka zdrojového kódu je pouze pro 10 ze 100 kontrolovaných komunikačních bajtů. Celý program obsahuje svou inicializační část, která má na starosti inkrementování aktuální hodnoty v intervalu 0 až 100. Poslední část, která vyhodnocuje pole „distance“ a zjišťuje kontrolu, zda v některém z komunikačních bajtů nebyla překročena hodnota rozdílu 2.

4.2.3 Projekt 5xP350

Dalším testem ve specifikaci je kontrola izochronní komunikace mezi 5 Simotion řady P350. Pro tento projekt jsou využity zkušenosti z předchozí konfigurace (2xD435). Zajištění kontroly komunikace je zde v principu stejné. Jelikož se jedná o větší počet výkonnějších kontrolérů, přistoupil jsem k návrhu sofistikovanějšího testu. Následující obrázek je kopií nastavení topologie. Jednotlivé P350 se obvykle označují jako PCStationX, PG/PC je konfigurační a kontrolní PC. Propojení mezi jednotlivými Simotion charakterizují fyzické spojení pomocí PROFINETU a zelené čtverečky jsou porty PNIOMC_Board 1 až 4 (zleva doprava).



Obr. 20 Navržená topologie pro projekt 5xP350

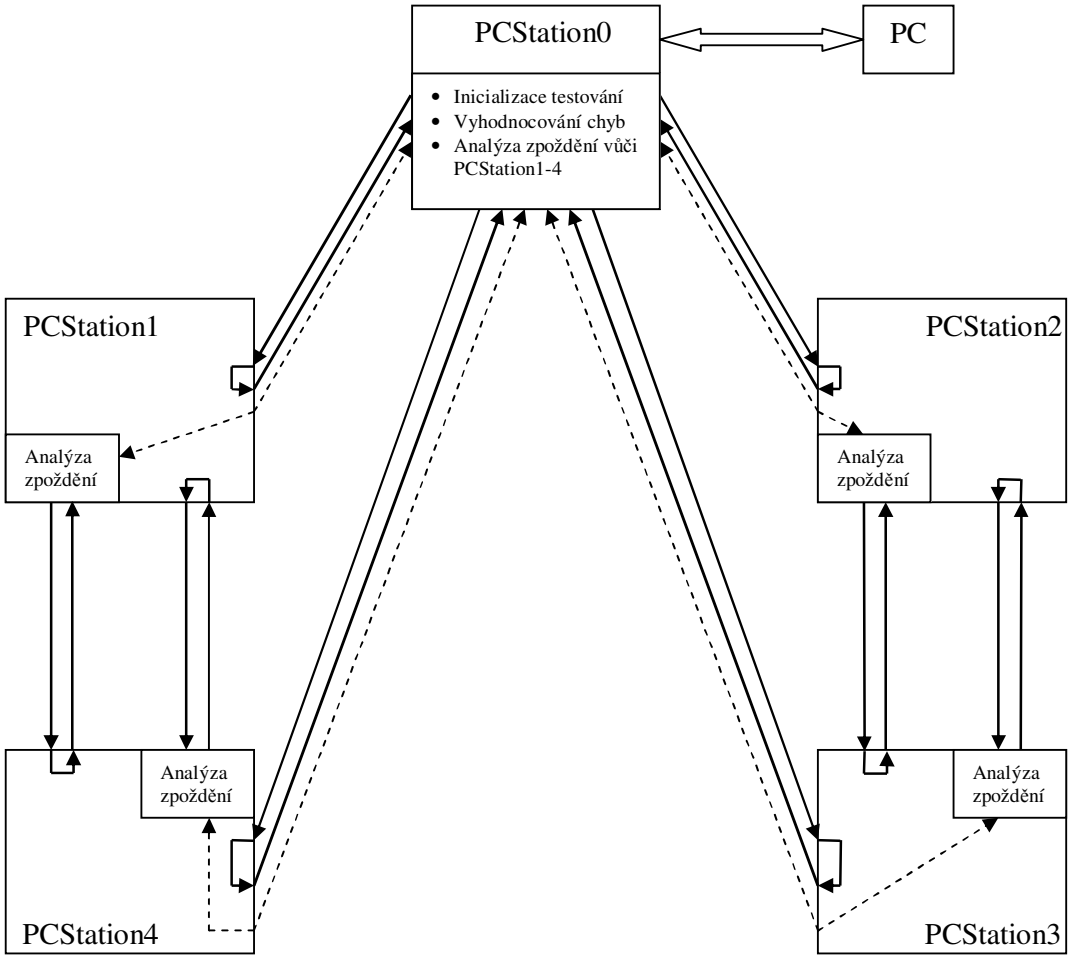
Na následujícím obrázku (obr. 21) je zobrazeno blokové schéma celého testovacího projektu. Z blokového schéma je patrné, že je zde jeden hlavní člen celého systému (PCStation0), který spouští a kontroluje celý test, vedlejší (podřízené) uzly obsahují testy pro větší komunikační zátěž.

Topologie a hlavní část testované komunikace je nastavena tak, aby jedním místem v topologii „teklo“ co nejvíce dat, konkrétně se jedná o kartu obsaženou v PCStation0. Mezi PCStation1 a PCStation4 je nastavena komunikace o délce 200 bajtů pro oba směry, taktéž mezi PCStation2 a PCStation3. Kontrola Watchdogu pro izochronní komunikaci zde probíhá stejně jako v předchozím projektu (2xD435). PCStation 1,2,3,4 obsahují stejný aplikační program, liší se pouze v nastavení komunikačních dat (nastavení kdo s kým). Hlavní Simotion P350 má nastavenou komunikaci o délce 20 bajtů s každou podřízenou P350. Těchto 20 bajtů zajišťuje kontrolu komunikace – test izochronnosti, taktéž data pro inicializaci a spuštění jednotlivých testů (na obrázku 21 jsou tyto dvě části charakterizované plnýma a čárkovanýma šipkama). Příkazy pro spuštění testu probíhají stylem příkaz ze strany hlavního kontroléru a potvrzení příkazu od podřízených kontroléru. Směrem k hlavnímu Simotion je posílána informace o počtu nalezených chyb během analýzy zpoždění pro hlavní část testu komunikace.

Celý test je ovládán z PCStation0, ke kterému je připojeno konfigurační PC. V parametrech testu je mimo jiné konfigurovatelná položka nastavení délky testu, po kterou nesmí dojít k rozsynchronizování komunikace v jakémkoliv místě. Samotné startování testu, vyhodnocování chyb a ukončení testu probíhá plně automaticky. Sledování průběhu testu může probíhat manuálně, případně automaticky pomocí externího testovacího skriptu z konfiguračního PC (více v 5. kapitole). Test stability komunikace probíhá po dobu 10 hodin s taktem 250 us.

V tomto projektu byl použit hardware:

- 5 x Simotion P350 verze 3 OČ: 6AU1 350-XXXXX-XXX2 / V4.1
- 5 x PNIOMC_Board (bez označení verze a objednávacího čísla)



Obr. 21 Blokové schéma projektu 5xP350

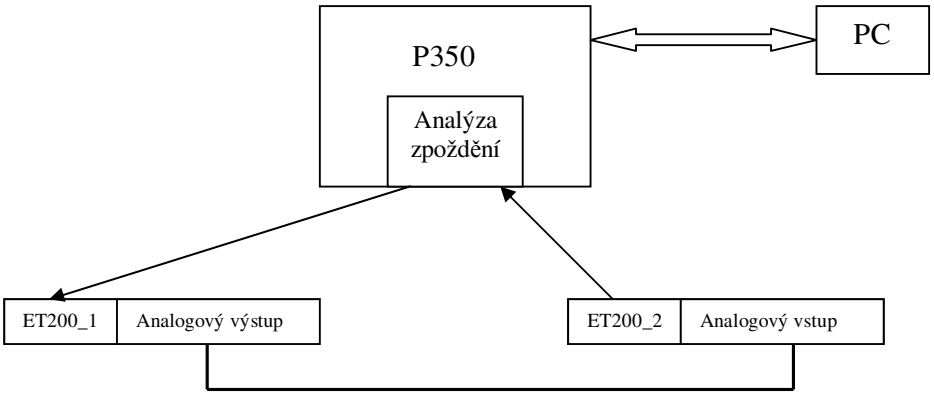
4.2.4 Projekt P350 + 2xET200

Posledním projektem popisovaným v této diplomové práci je test pro Simotion P350 a kontrola izochronní komunikace s moduly vzdálených vstupů ET200 s cyklem sběrnice 250 us. Požadavek na test jednotky ET200s High Speed byl kladen především na hlavu celé jednotky a nebyl svázán s konkrétními moduly. Hlava jednotky ET200 zajišťuje komunikaci se sběrnicí PROFINET a připojují se k ní jednotlivé moduly, třeba digitální vstupy/výstupy, analogové vstupy/výstupy. Jednotka ET200 je jedna z možností jak zajistit distribuovaným řídicím systémům čtení a zápis elementárních digitálních a analogových signálů.

Otázkou tohoto projektu je, jak zajistit kontrolu správného přenosu dat mezi Simotion a ET200. Jedna z možností je zapisovat a postupně inkrementovat hodnotu analogového výstup první jednotky ET200, propojení tohoto výstupu s jiným analogovým vstupem druhé jednotky ET200 a následná kontrola v přenosu informace v kontroléru.

V tomto projektu je použit tento hardware:

- 1 x Simotion P350 verze 3 OČ: 6AU1 350-XXXXX-XXX2 / V4.1
- 1 x PNIOMC_Board (bez označení verze a objednávacího čísla)
- 2 x ET200 IM151-3PN OČ: 6ES7 151-3BA60-0AB0
- 2 x Napájecí modul 24V DC OČ: 6ES7 138-4CA01-0AA0
- 1 x Analogový výstup HS OČ: 6ES7 135-4FB52-0AB0
- 1 x Analogový vstup HS OČ: 6ES7 138-4CA01-0AA0



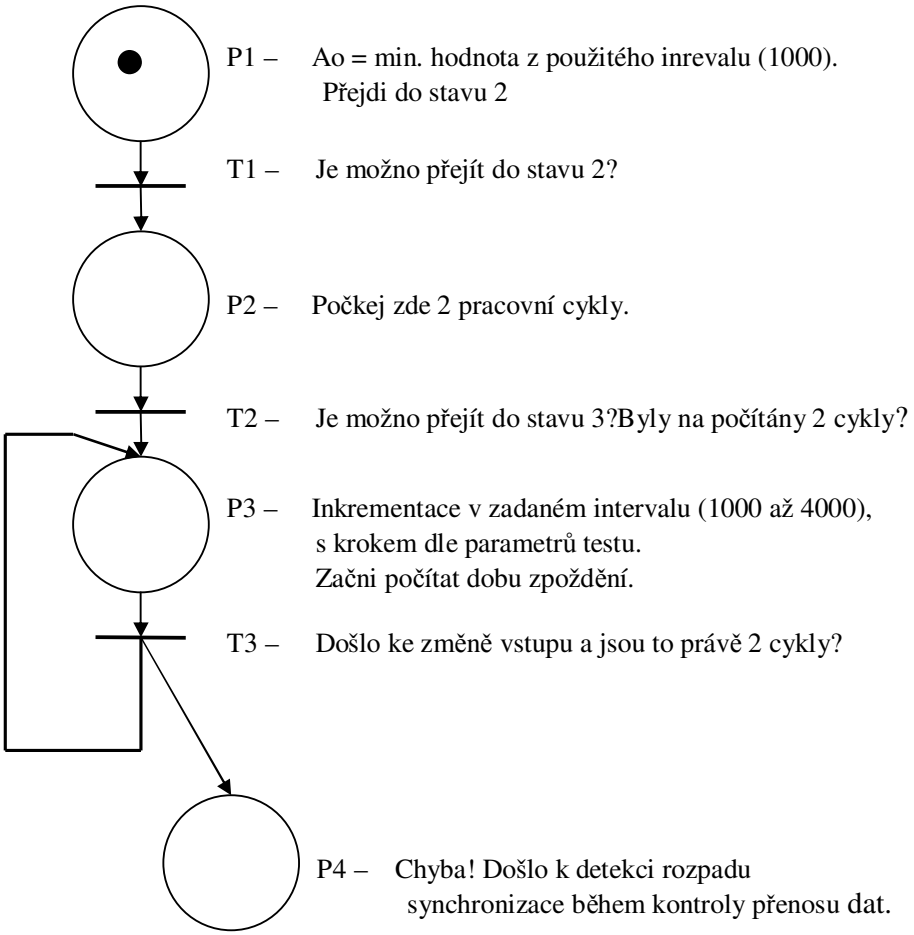
Obr. 22 Blokové schéma projektu P350 + 2xET200

Z předchozího blokového schématu je patrné, že komunikace od kontroléru je spjata s první ET200 (nazvaná ET200_1). Směrem k kontroléru přicházejí data od ET200_2. Princip kontroly izochronní komunikace je podobný té z předchozího projektu, ale nemůže zde být použita stejná aplikace.

O aktuální úrovni analogového vstupu, případně výstupu vypovídá 16 bitové číslo. Z toho je 15 bitů vyhrazeno samotnému převodu analogové hodnoty na digitální a jeden bit je zde pro určení případné polarity. Použité moduly nabízí možnost pracovat s napěťovými úrovněmi 1..5 V, +/-5 V, +/-10 V. Pro tento projekt je vybrána napěťová úroveň 1 až 5 V a to z důvodu možnosti diagnostiky zkratu (jedna z testovaných vlastností, není součástí této diplomové práce). Řetězec P350 – PROFINET – ET200 – AO – AI – ET200 – PROFINET – P350 musí vykazovat stejné zpoždění pro přenos dat, jako u předchozích projektů, a to zpoždění 2 pracovní cykly kontroléru. To znamená, že když v prvním cyklu zapíšu hodnotu na výstup, tak ve třetím cyklu v aplikaci přečtu ekvivalent zapsané hodnoty na vstupu. Velkým rozdílem oproti kopírování konkrétních ostrých čísel je, že analogové moduly nejsou standartně přesně kalibrovány a působí zde vnější vlivy. To neznamená, že by jejich měření nebylo přesné, ale zapsaná decimálně hodnota 1000 na analogový výstup neznamená, že se na propojeném analogovém vstupu přečte také decimální hodnota 1000. Při reškálovaní a škálovaní by se došlo k velice obdobné analogové hodnotě (např. při rozlišení 15 bity na rozsahu 1 až 5 V by hodnota 1000 byla 1,122V), ale decimálně se nebude jednat o úplně stejné číslo. Proto je nutné navrhnout testovací aplikaci tak, aby tuto vlastnost ošetřila.

Samotný testovací program má dvě části. Jednu inicializační, která je řešená jako stavový automat a druhou testovací, která inkrementuje hodnotu analogového výstupu a porovnává ji s analogovým vstupem. První část pracuje tak, že se v prvním stavu zapíše

na výstup minimální hodnotu z intervalu testovaných čísel (pro tento test se pracuje s hodnotami 1000 až 4000 decimálně s krokem 100, tyto hodnoty je možno změnit v parametrech testu) a přejde se do stavu dva. Ve druhém stavu se čeká 2 pracovní cykly na ustálení analogového vstupu. Po uplynutí 2 cyklů se uloží aktuální hodnota vstupu do pomocné proměnné a přejde se do třetího stavu, který již zajišťuje samotné testování izochronosti přenosu dat. Během testování komunikace se postupuje tak, že se zvedne hodnota výstupu o zvolenou inkrementační proměnnou a počítá se počet cyklů, než dojde ke změně na vstupu. Podmínka pro správnost testu je, že musí být počet takto napočítaných cyklů právě 2. Ke správnému vyhodnocení změny vstupu dojde tehdy, jestliže se vstup změní oproti poslední hodnotě vstupu o hodnotu inkrementování ± 30 % a následně dojde k uložení aktuální hodnoty vstupu pro příští porovnání.



Obr. 23 Petriho síť návrhu chování aplikace pro testování komunikace Simotion s dvěmi ET200

5 Automatizované testy

5.1 Možnosti automatizace

5.1.1 Možnosti k nasazení

Testování je nedílnou součásti vývoje každého nového produktu. Kvalita a kvantita testů daného produktu, před uvedením na trh, přímo ovlivňuje jeho spolehlivost. S rostoucím počtem implementovaných vlastností do systému Simotion a rozvojem komunikačního standartu PROFINET docházelo i k navyšování počtů testů a otevíraly se možnosti je automatizovat.

Při začátku vývoje nového produktu postačují tzv. „pudové“ testy. Nově vyvinutá vlastnost funguje nebo nefunguje. S rostoucím počtem těchto vlastností roste i počet elementárních testů, které je potřeba vykonat. Nové vlastnosti produktu většinou navazují na ty předešlé, ale v tak složitém systému, jakým je Simotion, může nově naimplementovaná vlastnost ovlivnit jinou, byť elementární, dříve testovanou. Proto některé testy je dobré navrhnout tak, aby zajistily testování více vlastností najednou a umožnily rychlou diagnostiku jednotlivých problémů – možnost automatizace.

Dalším otevřeným okruhem možné automatizace jsou časově náročné testy z hlediska četnosti provedení rutinních úkolů. Stejně jako v každém odvětví průmyslového vývoje, hraje důležitou roli zautomatizování jednoduchých procesů, jejichž vykonávání člověkem může vést ke zvýšení chybovosti. I v testování je tedy vhodné některé úkoly zautomatizovat. Časová náročnost některých testů trápila i integrační test při implementaci PROFINETU, a proto jsem přistoupil k návrhu některých automatických skriptů.

Před začátkem návrhu jakéhokoliv automatizovaného procesu je nutné si položit jednu otázku: Kolik času (peněz) mi automatizace ušetří? V mém případě je do odpovědi zahrnuta z jedné strany četnost vykonávání rutinních testů vynásobená jejich časovou náročností z dlouhodobého hlediska a na stranu druhou je postaven odhadovaný čas potřebný k vývoji automatizovaného testu, jeho údržbě a používání. Je naprosto zbytečné automatizovat některé procesy, které sice zaberou několik hodin s periodou několika dní, když se dopředu ví, že tento úkol není potřebné vykonávat dlouhodobě a čas potřebný k vývoji testu je větší, než čas potřebný k samotnému testování. Zato některé automatizované skripty bylo nutné rychle vyvinout, protože bylo zřejmé hned od začátku, že není v lidských silách tyto testy provádět. Jedním příkladem může být testování restartů kontroléru v násobcích tisíců cyklů. Samotný test může zabrat několik dní, jen doba náběhu kontroléru D435 se pohybuje od 30 do 60 sekund v závislosti na velikosti projektu. Proto je vývoj testů koncipován tak, aby se rychle dosáhlo zajištění produktivity v co nejkratším čase, ale současně se neuzavřely možnosti rozšíření systému na základě jeho modularity. Jestliže takový test začne generovat výsledky, rozvážou se ruce k jeho dalšímu vylepšování a adaptování na jiné projekty, případně pro distribuci jiným uživatelům.

5.1.2 Postup návrhu

Diplomová práce se omezuje na automatické testování vlastností, které jsou náročné počtem testovaných cyklů.

Okruhy k zautomatizování:

- Restartování kontrolérů
- Rozpojování kabelů mezi kontroléry
- Automatická konfigurace cyklu komunikace

První dva navržené testy v sobě koncipují zajištění elektrotechnické podpory pro testování, jako například napájecí relé a rozpojovače kabelů umožňující vzdálené ovládání. Touto částí se nebudu v diplomové práci dopodrobna zabývat, ale jen pro informaci uvedu, že pro rozpínaní napájení jsou použity standardní modulární relé firmy Shrack. Pro jejich ovládání jsou zakoupeny IO Kontroléry firmy HW Group, které obsahují 8 digitálních vstupů a výstupů (24V). IO Kontrolér je zařízení komunikující pomocí sítě Ethernet a ze strany testovacího PC musel být napsán jednoduchý ovladač v jazyce C# s programovým COM rozhraním, pro zajištění integrace do samotných skriptů testu. Pro rozpínání fyzické vrstvy PROFINETU (kabelu) je použita obdobná koncepce, ale relé jsou nahrazeny elektronickými spínači, které nevnášejí do datové komunikace takový šum, jaký by mohla vnášet relé.

Třetí automatizovaný test nepotřebuje ke svému chodu žádnou elektrotechnickou podporu, vše je zajištěno programově (automatické změny času kontroléru). Ale o to složitější problém nastává při hledání cesty jak tento test zajistit. Nakonec jsem přistoupil k využití interních funkcí programu Scout a Step7, ovládaných ze skriptovacího jazyka. Oba programy umožňují využití jejich interní funkcí bez použití grafického rozhraní. To znamená, že může být například vytvořen celý projekt, aniž bych otevřel klasické konfigurační prostředí. Pro tuto práci je třeba si vyrobit ve skriptovacím prostředí správné instance na správné objekty, které umožňují požadované vlastnosti. Pro automatickou změnu času je použita kombinace exportu ručně vytvořených projektů, programová konfigurace těchto exportovaných souborů, automatický import upravených exportovaných projektů a závěrečné nahrání celého projektu do kontrolérů.

Společným problémem všech tří projektů pro automatizované testování je diagnostika chybových stavů. Pro první dva projekty je nutná implementace diagnostického nástroje kontroly synchronizace do komunikačních karet (driver PROFINET). Komunikační karty CBE30 a PNIOMC_Board mají od samého začátku v sobě zakomponovanou diagnostiku a signalizace chyby synchronizace komunikace. Ale vnějšímu světu dávají tuto informaci najevo pouze pomocí dvou diod (červená a zelená). Trvalé svícení zelené diody znamená, že je vše v pořádku. Chybové stavy jsou signalizovány červenou diodou a to v několika stupních. Pro automatické testování je nutná strojově zpracovatelná diagnostika chyby synchronizace. Touto otázkou jsem se zabýval společně s vývojáři software a došli jsme k jednoduchému

řešení. V ovladači komunikační karty je zahrnuta služba telnet pro diagnostiku a konfiguraci. Pro indikaci komunikační synchronizace jsou dopsány pouze příkazy, které vracejí pouze informaci funguje-nefunguje k předešlé implementaci telnetu. Pro zajištění funkce telnetu je napsána jednoduchá aplikace v programovém jazyce C#, která je pak využita ve skriptovacím jazyce (viz další kapitola).

Diagnostika chybových stavů pro automatickou změnu času je zajištěna dvojím způsobem. První je test synchronní komunikace na nejnižší úrovni stejně jako v předešlém případě pomocí telnetu. Druhý stupeň je na úrovni aplikace v kontroléru. Především pro nejkratší cykly komunikace se stávalo, že ne všechna data doputovala ke svému cíli v definovaném čase. To se projevuje tak, že v testování zpoždění komunikace došlo k výpadku jedné hodnoty a jiná se zase objevila dvakrát. Tuto chybu dokáže diagnostikovat například projekt 5xP350 uvedený v předešlém textu. Problémem však je zajištění komunikace automatického skriptu s aplikací vykonávanou v kontroléru Simotion. Pro tuto komunikaci jsou využity interní funkce konfiguračního prostředí Scout, které umožní, stejně jako v grafickém rozhraní, se připojit do stavu online ke kontroléru. Ve stavu online je možno přepnout kontrolér do RUN, zapisovat a číst z proměnných programu. Při vhodném navrhnutí aplikace kontroléru pro roztáčení motorů lze i diagnostikovat synchronní aplikace pomocí elektronických převodovek.

Pro zvýšení univerzálnosti a zajištění snadného používání testovacích skriptů je navrhnuto specifické konfigurování skriptů. Kontraproduktivní by bylo psát skript přesně na míru danému projektu. Proto jsem přistoupil k návrhu konfiguračních souboru, které definují jaká zařízení se používají (společně s jejich parametry), jaké jsou mezi nimi vztahy (kdo je synchronizační master, slave) a v jaké topologii jsou zapojená.

Samotná logika testů byla pak realizována ve skriptovacím jazyce Python. Průběh testů se vypisuje na příkazový řádek a současně se zapisuje do logovacích souborů.

Pro zvýšení využití testovaných zařízení v době pracovního volna bylo dobré vymyslet možnosti dávkového spouštění. Tyto možnosti rozeberu v následujícím textu.

Aktuální stav automatizovaných testů v sobě zahrnuje tyto moduly:

- Elektrotechnická podpora (relé, rozpojovače kabelů)
- Diagnostika synchronní komunikace na úrovni komunikačních karet PROFINET
- Integrace interních funkcí Scout a Step7
- Konfigurační soubory
- Zdrojové kódy skriptů
- Soubory s výsledky testů
- Možnosti dávkového spouštění

5.1.2 Rozbor okruhů k zautomatizování

Okruhy automatických skriptů pro tuto diplomovou práci jsou celkem tři. První z nich se zabývá automatickým testováním co největšího počtu restartů kontrolérů v projektu MAN. Pro připomenutí, tento projekt simuluje tiskařský stroj firmy Man Roland a celkově je zde použito 13 kontroléru typu Simotion D435. Mezi Simotion je navázána izochronní komunikace s cyklem 1 ms (RT_Class3, PROFINET IRT). První kontrolér je synchronizační master, ostatní jsou slave. Principem testování restartu těchto zařízení je, že jakýkoliv z kontrolérů může být kdykoliv odpojen od napájení 24V. Po opětovném zapnutí musí do 70 sekund dojít k navázání synchronní komunikace pro všechny Simotion. Pro kontroléry řady D435 by mělo dojít k nastartování do jedné minuty, po kterém může dojít k navazování komunikace přes PROFINET. Pro takto velký projekt trvá navázání výměny dat cca 10 sekund. Tyto časy nejsou výrobcem specifikovány, ale po domluvě s německými kolegy byl stanoven jako hraniční čas náběhu 70 sekund, který by neměl být překračován. Po navázání komunikace může dojít k roztočení motorů, tento test kontroluje jen správné nastartování PROFINETU. Testování roztočení motorů po restartu je proveditelné. Pro zvýšení produktivity testování navázaní samotné komunikace je upuštěno od kontroly točení motorů. Pro zajímavost uvedu, že po odpadu napájení například 3 kontrolérů (v liniové topologii) dojde k rozpadu komunikace pouze se Simotion, které jsou zapojené za tímto restartovaným zařízením. V praxi to znamená, že motory připojené ke kontrolérům před tímto vypnutým zařízením se budou stále točit. Test simuluje chybové stavy, které mohou nastávat v důsledku výpadku napájení, a to v největší možné kvantitě za nejkratší čas (v jeden čas se vypíná pouze jeden kontrolér).

Druhý automatizovaný test se zabývá obdobnou problematikou jako test první simulace chybových stavů. Místo restartů se zde testuje rozpojování kabelů. Na rozpojování kabelů je z historického hlediska kladen větší důraz, především na kvantitu. K rozpojení kabelů, dle zkušenosti z technické praxe, může dojít častěji. Přece jen konektor RJ45, používaný pro fyzickou vrstvu Ethernetu a PROFINETU, je snadno vypojitelný, nebo může dojít k velice krátkému výpadku komunikace (jen část komunikačního rámce). Proto zde nastává větší tlak na správné naprogramování všech driveru celého systému. Po restartu dochází k navázání komunikace tzv. načisto. Ale po rozpojení kabelu je nutné správně ukončit komunikaci uvnitř kontroléru a připravit se na navázání nové a to ve zlomku sekundy. Často se pak může stát, že nedojde ke správnému dealokování používané paměti v komunikační kartě CBE30 a může dojit k fatálnímu erroru z hlediska ovladače PROFINETU. Proto každá verze firmware musí být podrobně testována na rozpojování kabelů až v několika desítkách tisících cyklů.

Poslední test zapadá do okruhu automatické konfigurace systému. A to z hlediska změny času komunikace. Konkrétně pro kontroléry řady P350 je možné konfigurovat periodu komunikace v rozsahu 250 us až 4ms. Samotná změna času není z hlediska projektu složitá, ale je časově náročná. Při kroku 125 us je nutné celý projekt 31 krát změnit, zkompilovat, nahrát a spustit (naváže se správná komunikace). Tento proces ručně trvá asi 3 hodiny pro projekt 5xP350. Celý test je nutno provádět pro každou verzi firmware a proto jsem automatizoval

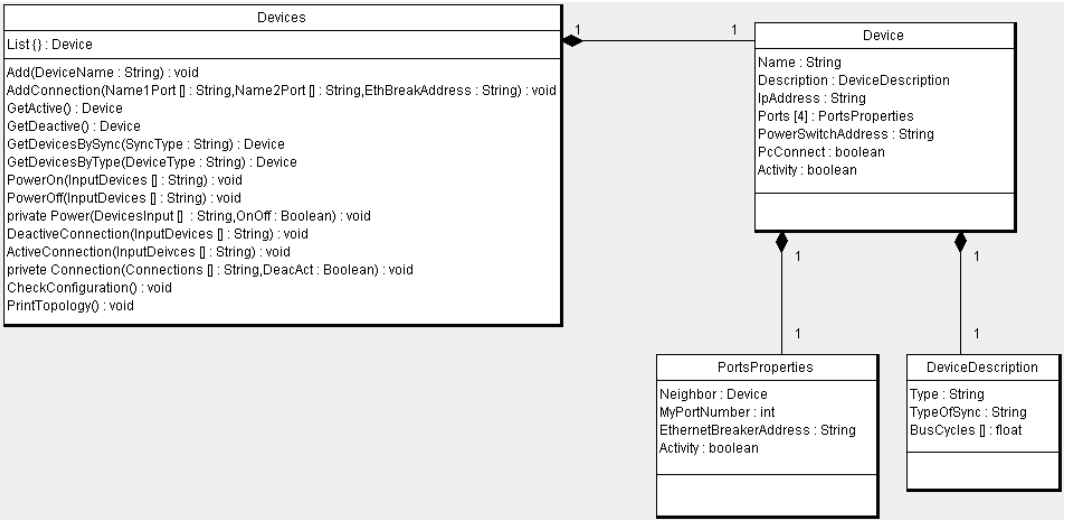
i tento test, i když se jedná o nejobtížněji proveditelný test. Automatizace umožňuje i zvýšení doby, po kterou se kontroluje správná výměna dat.

5.2 Realizace

5.2.1 Popis modulů

Konfigurační modul

Prvním popisovaným modulem, který využívají všechny automatické testy je modul nazvaný Devices_CL. Slouží ke konfiguraci parametrů testů. Je napsaný ve skriptovacím jazyce Python a nejlépe ho přiblíží následující obrázek – class diagram.



Obr. 24 Diagram tříd modulu Device_CL

Tento modul obsahuje celkem 4 třídy:

- Devices – Hlavní třída obsahující metody potřebné pro konfiguraci zařízení pro test a jejich seznam
- Device – Třída nahrazující datovou strukturu pro definici vlastností testovaných zařízení
- PortsProperties - Třída nahrazující datovou strukturu pro definici vlastností portů a spojení mezi zařízeními
- DeviceDescription - Třída nahrazující datovou strukturu pro definici podrobných vlastností testovaných zařízení

Cílem vývoje tohoto modulu bylo zjednodušení a zpřehlednění zadávání konfiguračních dat pro automatické testy. V praxi to pak vypadá, že jeden automatický test si vždy načte správný konfigurační soubor a samotnému test je už pak jedno, jestli testuje restarty dvou

kontrolérů nebo dvaceti kontrolérů. Tento modul dokáže podchytit i stromovou topologii. Jednotlivé vlastnosti, které umožňuje nastavit tento konfigurační modul, přiblíží následující text.

Proměnné

Name – Jméno zařízení. Musí být jedinečné.

Description – Vlastnosti zařízení, které jsou definovány strukturou následujících proměnných:

- **Type** – Nastavení typu zařízení (Simotion D435, P350, D445, ale i třeba Sinamics S120). Podle typu zařízení se pak může automatický test rozhodnout, jakým způsobem dané zařízení testovat. Toto rozlišení je především pro rozdílný přístup diagnostiky synchronizace mezi Simotion a Sinamics. Sinamics je zařízení ovládané Simotion, není v této diplomové práci použito.
- **TypeOfSync** – Rozlišení druhu synchronizace (RT, IRT master, IRT slave).
- **BusCycles** – Násobky nastavitelného času komunikačního rozhraní. Kontroléry se liší svým minimálním časem sběrnice (záleží na výkonu), i krokem nastavitelného času.
- **Neighbor** – Soused komunikačního spojení. Je datového typu Device – reference na sousední zařízení v seznamu.

IpAddress – IP adresa zařízení.

Ports – Pole o délce 4 definující vlastnosti spojení jednotlivých portů je charakterizováno následující strukturou:

- **MyPortNumber** – Číslo portu pro definované spojení (1-4)
- **EthernetBreakerAddress** – Adresa rozpojovače kabelu, nemusí být zadána – spojení bez rozpojovače. Je charakterizována IP adresou rozpojovače a pinem, na kterém je připojeno elektronické relé.
- **Activity** – Aktivita spojení (zapnuto, vypnuto).

PowerSwitchAddress – Adresa umožňující restart vypnutí napájení zařízení, opět je definována IP adresou IO Kontroléru a pinem konkrétního relé.

PcConnect – Proměnná definující, zda je k tomuto zařízení připojen testovací počítač, který vykonává automatický test. Tento údaj je velice důležitý, umožňuje rozklíčování problému s restarty a rozpojování kabelů v jakékoliv topologii. Příklad: je daná liniová topologie, jsem připojen k prvnímu zařízení a vypnu ho. V tento okamžik nejsem schopen komunikovat s žádným zařízením a nejde o chybu.

Activity – Aktivita zařízení (zapnuto, vypnuto).

List – Dynamický seznam nakonfigurovaných zařízení.

Metody

Add – Slouží pro přidání zařízení do seznamu.

AddConnection – Konfigurace spojení.

GetActive – Vrací aktivní zařízení. Pro určení aktivity je zohledněno jejich zapnutí nebo vypnutí, ale také jestli je komunikační spojení aktivní směrem od počítače k zařízení.

GetDeactive – Vrací neaktivní zařízení.

GetDevicesBySync – Filtr zařízení podle nakonfigurovaného typu synchronizace.

GetDevicesByType – Filtr zařízení podle nakonfigurovaného typu.

PowerOn – Zapne zvolená zařízení. Výběr je realizován podle jména.

PowerOff – Vypne zvolená zařízení. Výběr je realizován podle jména.

ActiveConnection – Zapne zvolená spojení. Výběr je realizován podle jména spojení.

DeactiveConnection – Vypne zvolená spojení. Výběr je realizován podle jména spojení.

CheckConfiguration – Zkontroluje správnost nastavení. V tomto testu jsou postupně vypínána všechna zařízení a spojení. Test dokáže zkontrolovat správnost IP adres zařízení a nastavení všech vypínacích prvků. Celá funkce je optimalizována pro co nejkratší časovou náročnost, a to za pomocí postupného hledání zařízení z konce topologie vzhledem k testovacímu počítači.

PrintTopology – Tato metoda není implementována, v budoucnu má tisknout topologii na obrazovku.

Následující ukázka zdrojového kódu konfiguračního souboru s komentářem přiblíží použití tohoto modulu pro dva Simotion D435.

```
#Import souboru, ktery obsahuje konfiguracni modul
from Devices_CL import *

#Vytvořeni instance třídy Devices
Devices = Devices()

#Přidáni zařízeni do seznamu
Devices.Add("D435_0")
Devices.Add("D435_1")


#Definice typu synchronizace
#IRT_SS - Sync Slave
```

```
#IRT_SM - Sync Master
#IRT_RSM - Redundant Sync Master
Devices.TypeOfSync = ["RT","IRT_SS","IRT_SM","IRT_RSM"]

#Definice typu zařízeni
Devices.TypeOfDevice = ["D435","S120"]

#Definováni vlastnosti prvního kontroléru
Devices.List["D435_0"].Description.Type          = Devices.TypeOfDevice[0]
Devices.List["D435_0"].Description.TypeOfSync    = Devices.TypeOfSync[2]
Devices.List["D435_0"].Description.BusCycle      = [0.5,1,1.5,2,2.5,3,3.5,4]    #ms
Devices.List["D435_0"].IpAddress                 = "192.168.0.10"
Devices.List["D435_0"].PowerSwitchAdrress        = "192.168.200.112:0"
Devices.List["D435_0"].Activity                  = 1
Devices.List["D435_0"].PcConnect                 = 1

#Definováni vlastnosti druhého kontroléru
Devices.List["D435_1"].Description.Type          = Devices.TypeOfDevice[0]
Devices.List["D435_1"].Description.TypeOfSync    = Devices.TypeOfSync[1]
Devices.List["D435_1"].Description.BusCycle      = [0.5,1,1.5,2,2.5,3,3.5,4]    #ms
Devices.List["D435_1"].IpAddress                 = "192.168.0.11"
Devices.List["D435_1"].PowerSwitchAdrress        = "192.168.200.112:1"
Devices.List["D435_1"].Activity                  = 1
Devices.List["D435_1"].PcConnect                 = 0

#Definice spojeni mezi kontroléry port 2 na D435_0 a port 1 na D435_1 s použitym rozpojovačem kabelu
#V případě, že spojeni není nakonfigurováno, zařízení není v testu použito i všechny další, které se na něj odkazuji svou topologii
Devices.AddConnection(["D435_0",2],["D435_1",1], "192.168.200.105:0"))
```

Výpis zdrojového kódu pro konfigurační soubor

Takto definovaný konfigurační soubor je pak naimportován do samotného automatizovaného skriptu a všechny jeho vlastnosti mohou být dále využity.

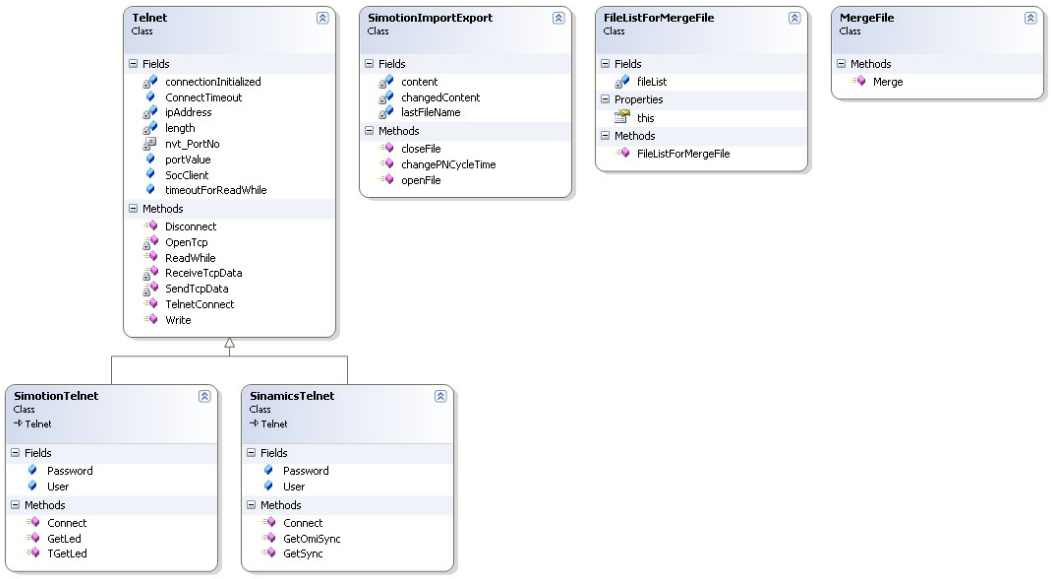
Modul Tests.dll

Následující modul popsaný v této diplomové práci je nazvaný Tests.dll. Jedná se o knihovnu tříd a metod sdružující několik funkcí automatizovaných testů. Celá knihovna je napsána v jazyce C# a je zapouzdřena COM rozhraním, které umožňuje integraci jejich vlastností do kteréhokoli programovacího jazyka, jež dokáže referencovat knihovnu s COM rozhraním. Jazyk C# byl vybrán pro realizaci některých funkcí z toho důvodu, že dokáže snáze pracovat se síťovým rozhraním Ethernet než jazyk Python na úrovní TCP/IP. I metody pro práci se soubory jsou lépe zpracované v .Net framework (pracovní prostředí, které využívá programování ve Visual Studiu a jazyk C#) než je tomu v knihovnách, které jsou k dispozici v jazyce Python.

Knihovna Test realizuje tyto dílčí úkoly automatizovaných testů:

- Změna exportovaných souborů z HW Config – úprava cyklu komunikace - **SimotionImportExport**
- Realizace telnetu pro testování synchronní komunikace – **Telnet**
- Spojování souboru pro dávkové spouštění testů – **MargeFile**

Knihovnu Tests nejlépe přibližuje následující obrázek, ve kterém je znázorněn diagram tříd celého modulu. Tento diagram tříd byl vygenerován s Visual Studia 2008.



Obr. 25 Diagram tříd modulu Tests.dll

Prvním realizovaným zadáním modulu je nástroj pro změnu exportovaných souborů z prostředí HW Config. HW Config je součástí prostředí Step7 a v systémech Simotion zajišťuje konfiguraci použitého hardwaru a následné plánovaní komunikace (v mém případě PROFINET IRT). Základním problémem automatické změny času bylo to, že prostředí HW Config nemá zveřejněnou žádnou funkci, která umožňuje externímu nástroji měnit nastavení komunikačního cyklu. Ale umožňuje externí importy konfiguračního souboru. Proto je přistoupeno k takovému řešení, že se ručně vytvořil projekt, ze kterého se pak exportuje konfigurační soubor celého HW Configu. Metodou porovnání dvou souborů jsem přišel na to, na kterých místech jsou uloženy informace o nastaveném cyklu použitých zařízení a pak jsem vytvořil program, který podle nastudovaných pravidel dokáže měnit exportovaný soubor.

Další realizovanou funkcí modulu Test je třída pro Telnet. Třída nazvaná Telnet realizuje samotnou vrstvu Telnetu dle jeho všeobecných specifikací (TCP/IP protokol, komunikace pomocí Socket, port 23). V .Net framework jsem nenašel žádné třídy, které dokážou realizovat funkci telnetu, proto jsou všechny metody navrženy a realizovány mnou. Využil jsem proto systémových funkcí System.Net.Sockets. Základní vlastnosti telnetu v mém modulu dědí další dvě třídy nazvané SimotionTelnet a SinamicsTelnet. Tyto dvě třídy pak v sobě obsahují funkce napsané na míru funkcím komunikačních karet CBE20 a CBE30. Pro zjištění zda-li je PROFINET v kontroléru synchronní slouží funkce GetLed, která vrací OK nebo NOK. Což je pro automatické testy dostačující.

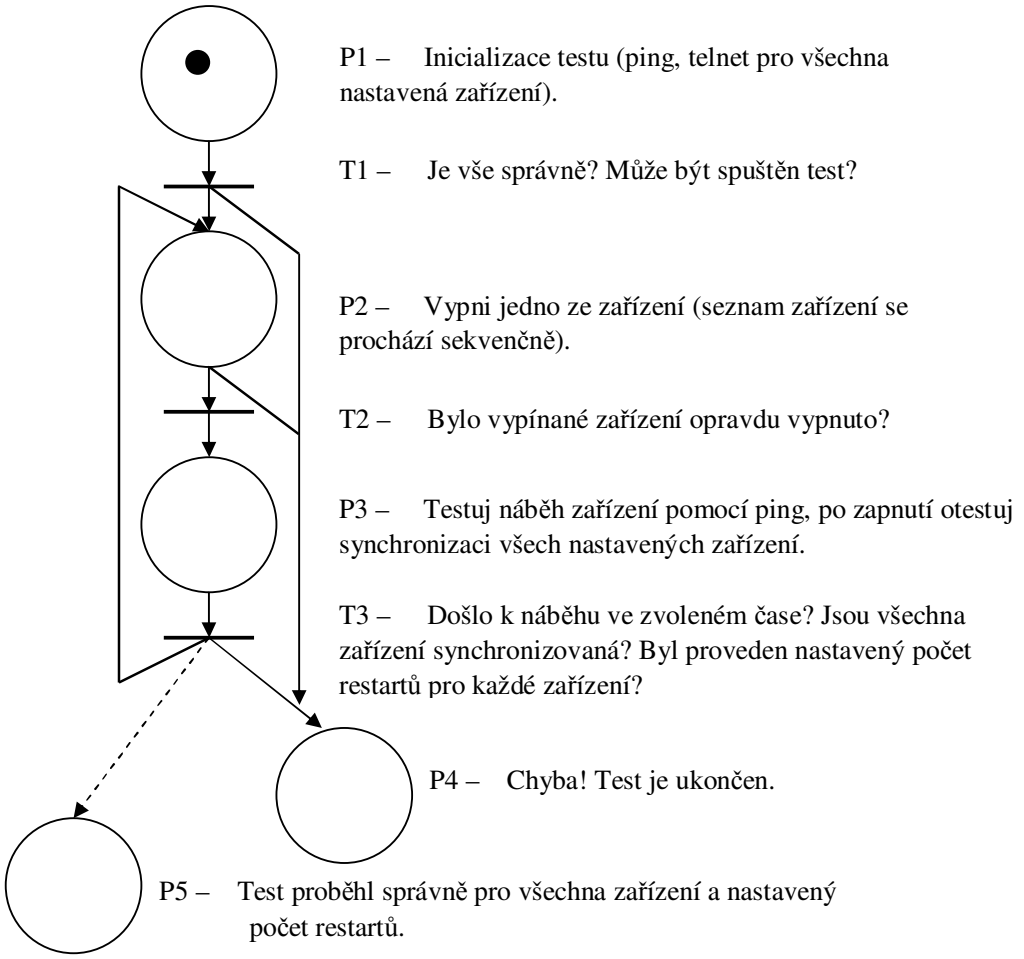
Poslední realizovanou funkcí modulu Tests je nástroj pro dávkové spouštění testů. V této knihovně je problém řešen velice jednoduše. Pomocí metody Marge je spojen zdrojový kód automatických testů do jednoho souboru, který pak může být spuštěn. Je to stejné, jako by se zdrojový kód zkopíroval ze dvou souborů do jednoho. Toto řešení dovoluje spojování více nezávislých testů do jednoho za předpokladu, že v jednu dobu může běžet jen jeden test na jednom testovacím počítači. Tato funkce je využívána především v době pracovního volna, před kterým je takový test připraven a postupně například během víkendu se může provést několik dílčích testů. Třída FileListForMargeFile umožňuje vytvářet spojitý seznam souboru ke spojení. Předávání pole pomocí COM rozhraní není snadno vyřešitelné a vytvoření nové třídy je řešením, jak tento problém obejít.

5.2.2 Skripty automatizovaných testů

Restarty kontrolérů - MAN

Testování restartů zařízení je realizováno v jazyce Python. Tento skriptovací jazyk umožňuje poměrně snadné psaní kódu s grafickým výstupem na příkazovou řádku. Všechny testy jsou řešeny podobně jako stavový automat. Sekvence podmínek a úkonů jsou napsány tak, jako by se test prováděl ručně. Nejprve je ověřen výchozí stav (vše je zapnuto a funguje správně), poté dojde k simulaci chybového stavu (je vypnuto jedno se zařízení) a navrácení do původního stavu (zařízení je zapnuto). Po náběhu zařízení je opět ověřena správnost komunikace (všechna zařízení jsou synchronizována pomocí PROFINET), poté je opět simulován chybový stav atd.

Celý test je rozdělen do několika souborů, konfigurační soubor (zde je využit konfigurační modul), zdrojový kód testu, parametrizační soubor (nastavení počtu restartů, časový limit pro náběh zařízení) a pak spouštěcí soubor, kde může být využita funkce pro spojování zdrojových kódů. Zdrojový kód pro restartování simulace tiskařského stroje je napsán univerzálně tak, aby mohl testovat jak kontroléry Simotion, tak i Sinamics device. V této diplomové práci se zabývám pouze kontroléry, proto specifika testování zařízení Sinamics nebudu rozebírat. Průběh testu je vypisován na příkazový řádek a současně jsou všechny informace zapisovány do logovacího souboru. Test se nezajímá o aplikační vrstvu systému Simotion (netestuje točení motoru a výměnu dat), ale pouze o synchronizovaný PROFINET. Filozofii automatického testu nejlépe přiblíží následující obrázek.



Obr. 26 Petriho diagram znázorňující průběh automatizovaného testu pro restarty zařízení

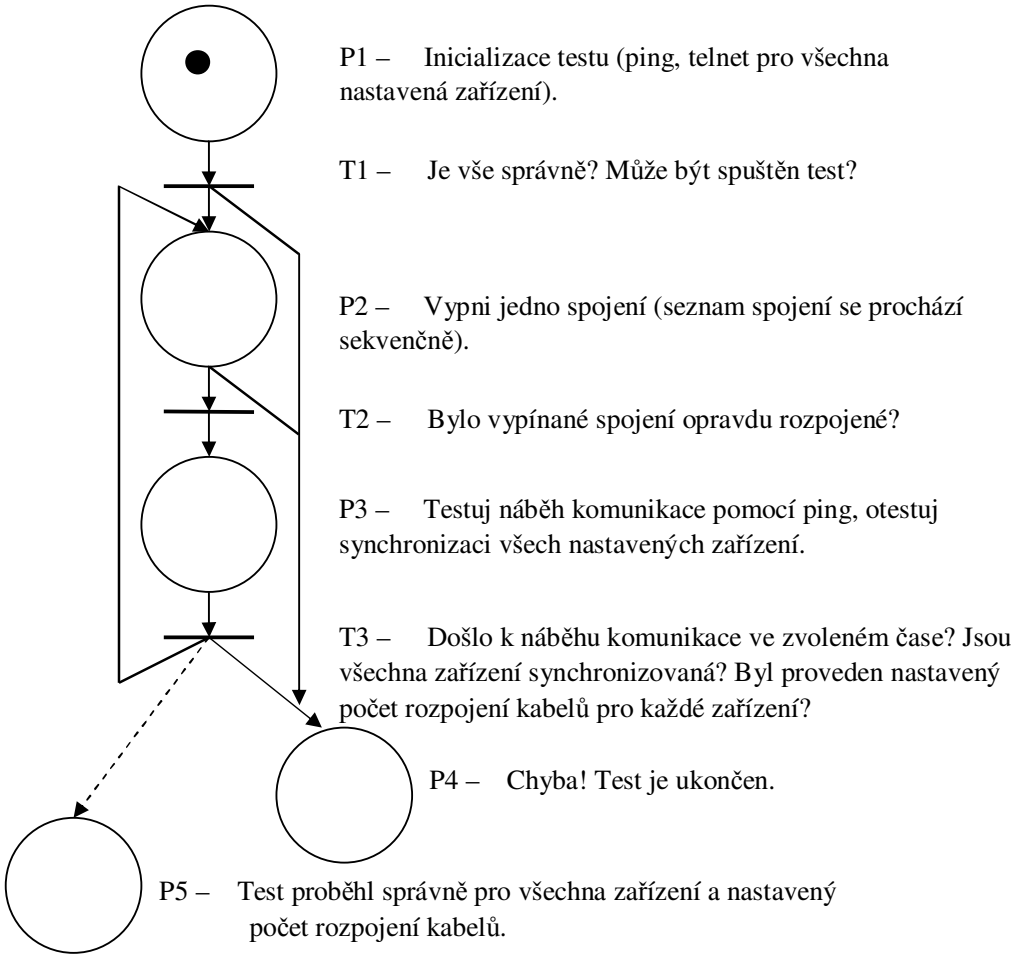
```
# Vypnuti zařízení
Tisk.InfoMessage("Switching OFF relays...")
PowerOff.Outputs[RelayPort[TestStation]] = 1
time.sleep(2)

# Testování vypnuti
Tisk.InfoMessage("Testing if station is off...")
if (TestStation < NoControllers):
   StationIP = testPy.inc_ip(AoControllers, TestStation)
else:
   StationIP = testPy.inc_ip(AoDevices, TestStation - NoControllers)
PingOK = testPy.pingTest(StationIP)
if PingOK:
    Tisk.FaultMessage("Station %s not off!!!" % StationIP)
    TestOK = 0

# Zapnuti zařízení
Tisk.InfoMessage("Switching ON relays...")
PowerOff.Outputs[RelayPort[TestStation]] = 0
```

Výpis zdrojového kódu restarovacího skriptu (vypnutí a zapnutí kontoléru)

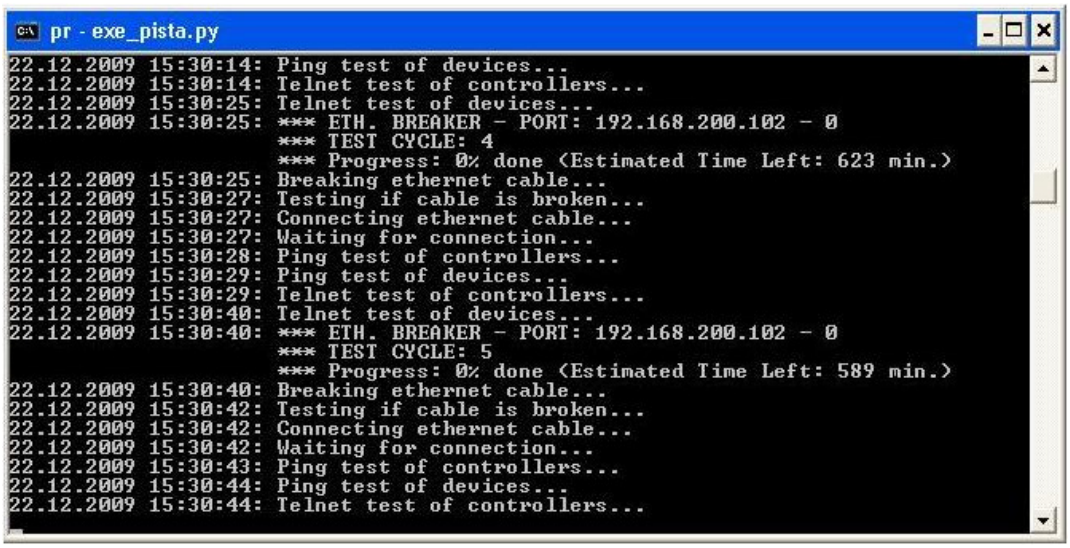
Ukázka souboru s výsledky průběhu testu je umístěna v elektronické příloze této diplomové práce (totožné s tím, co se vypisuje na obrazovku v průběhu testu).

Rozpojování kabelů – 5xP350

Testování rozpojování kabelů pro projekt 5xP350 vychází ze stejné filozofie jako testování restartů. Vše je založeno na simulaci a kontrole tak, jak by se celý test dělal manuálně. Zdrojový kód pro tento test je také velice obdobný, jen místo restartu se vždy provede rozpojení spojení mezi dvěma kontroléry.



Obr. 27 Petriho diagram znázorňující průběh automatizovaného testu pro rozpojování kabelů



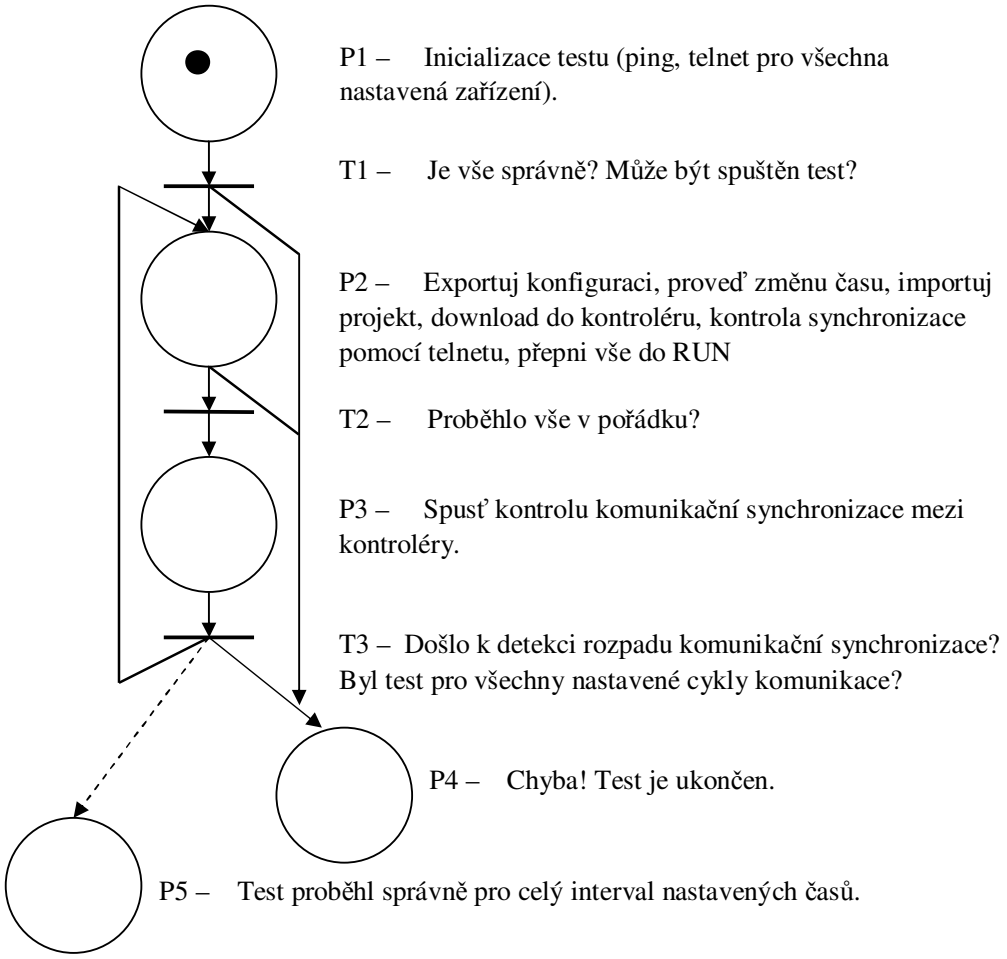
Obr. 28 Screenshoot obrazovky z průběhu automatického testu pro rozpojování kabelů

Ukázka souboru s výsledky průběhu testu je umístěna v elektronické příloze této diplomové práce (totožné s tím, co se vypisuje na obrazovku v průběhu testu).

Automatická změna komunikačního cyklu – 5xP350

Poslední popisovaný test se realizuje automatickou konfigurací cyklu komunikace mezi 5 kontroléry P350. Tento skript je podobně koncipován jako oba předešlé. Je napsán v jazyce Python, využívá modulu Tests.dll, ale jeho hlavní část využívá interních funkcí konfiguračního prostředí Scout a Step7. Využití těchto funkcí přináší jedno velké úskalí a to, že publikované funkce konfiguračních nástrojů nejsou primárně určeny pro takovéto použití. A proto ne všechny fungují tak jak by měly a celý test musel být tomuto faktu uzpůsoben. Oproti předchozím skriptům je po změně času otestovaná komunikace na aplikační úrovni. To znamená, že je na určitý čas spuštěn test, který popisuje kapitola 4 této diplomové práce, a je vyhodnoceno testování zpoždění komunikace mezi kontroléry pro každý komunikační cyklus.

Jádro automatické konfigurace se skrývá v knihovně project_class, která využívá interních funkcí Scout a Step7, tiskne na obrazovku, testuje synchronizaci pomocí telnetu a jednotlivé částí testu zabaluje do funkcí. Tyto funkce jsou pak postupně spouštěny z hlavního programu. Funkci celého testu přiblíží následující Petriho diagram funkce a ukázka zdrojového kódu.



Obr. 29 Petriho diagram znázorňující průběh automatizovaného testu pro změnu cyklu komunikace

```
#Začátek jednoho testovacího cyklu
Test_cycle = Test_cycle + 1
Tisk.InfoMessage("*** SEND CLOCK CHANGE TEST - CYCLE: %s"%Test_cycle)

#Cyklus pro zmenu konfiguračních souborů pro všechny kontroléry
for Index, ControllerName in enumerate(Controllers) :

    #Export konfiguračního souboru
    ExportFileCfgStationS7forChange      = ExportDir + "XML_P350" + "_" + str(Index) + "(Station)\\S7Station\\S7Station.cfg"
    Tc_clock_index,Tc_clock              = SimotionProject.Clock_pick (Clock_array,Test_cycle)

    #Změna konfiguračního souboru
    Send_clock_ret_data                  = SimotionProject.Send_clock(Tc_clock_index,ExportFileCfgStationS7forChange)

    #Import základního projektu
    ImportFileXml                    =  ExportDir + "P350" + "_" + str(Index) + "(Station).xml"
    Import_config_ret_data               =  SimotionProject.Import_config_P350x5 (ControllerName,ImportFileXml)
```

```
    #Kontrola a build
    Project_build_ret_data          = SimotionProject.Project_build ()

#Vyber kontroléru pro mód online
Controller_select_ret_data      = SimotionProject.Controller_select ()

#Přechod do stavu Online
Project_online_ret_data         = SimotionProject.Project_online()

#Přepnuti všech kontroléru do STOP
Controller_stop_ret_data        = SimotionProject.Controller_stop()

#Nahráni projektu do kontroléru - Download
Project_download_ret_data       = SimotionProject.Project_download ()

#Kopie RAM2ROM
Controllers_R2R_ret_data        = SimotionProject.Controller_RAM2ROM ()

#Přepnuti do Run modu
Controller_run_ret_data         = SimotionProject.Controller_run()

#Start testovací aplikace
Start_TC_ret_data               = SimotionProject.Start_TC  ()

#Kontrola testovací aplikace
Scout_test_result,System_clock_scout  = SimotionProject.Test_run(Tc_active_time)

#Zápis výsledku do logovacího souboru
Results2file_ret_data           = SimotionProject.Results2file(Scout_test_result,System_clock_scout,ResultsDirectory,Test_cycle)

#Přepnuti do Stop modu
Controller_stop_ret_data        = SimotionProject.Controller_stop()

#Projekt Offline
Project_offline_Ret_Data        = SimotionProject.Project_offline()
```

Výpis zdrojového kódu pro automatickou změnu času v projektu 5xP350 (sekvence příkazů využívající knihovnu project_class)

Ukázka souboru s výsledky průběhu testu je umístěna v elektronické příloze této diplomové práce (totožné s tím, co se vypisuje na obrazovku v průběhu testu).

6 Závěr

Diplomová práce se zabývá realizací požadavků integračního testu při vývoji a implementaci komunikačního rozhraní PROFINET do systémů Simotion. Výsledky této diplomové práce jsou v praxi používány a svým podílem přispěly k zvýšení spolehlivosti nového produktu společnosti Siemens.

Výchozím stavem pro zpracování této práce byly pouze požadavky na navržení vhodných testů nově implementovaných vlastností. Proto bylo nutné nastudovat způsoby konfigurace systémů Simotion, vlastnosti komunikačního rozhraní PROFINET, fyzicky sestavit testovací soustavy a realizovat samotné testy. Během testování jsem došel k řadě vylepšení postupů testování a v diplomové práci prezentuji již finální výsledky. Velikou otázkou integračního testu byla automatizace testů, jejichž možnosti a realizaci popisuje tato práce.

Samotná diplomová práce rozebírá základní vlastnosti komunikačního rozhraní PROFINET a systému Simotion. Poté následuje specifikace testovaných vlastností, možnosti realizace testů a samotný návrh řídicích aplikací pro několik konfigurací. Testovací aplikace jsou popsány pomocí blokových schémat, ukázek zdrojových kódů a slovního popisu. Pro zvýšení produktivity testování a realizaci testů, které by nemohly být prováděny manuálně, jsem navrhl několik automatických testů. Tyto automatické testy se zabývají simulací chybových stavů a automatickou konfigurací systému. Pro zajištění správné funkce těchto testů jsem navrhl konfigurační modul a modul, který zajišťuje některé elementární úkoly. Samotné automatické testy jsou rozebrány v práci pomocí Petriho sítí, ukázek zdrojového kódu a slovního popisu.

Výsledky mé práce můžou být dále rozvíjeny dle aktuálních požadavků nově implementovaných vlastností. V krátkodobém horizontu budou automatické testy rozšířené o možnosti dynamické změny topologie za pomocí použití většího počtu rozpojovačů kabelů a konfiguračních souborů. Dále bude do modulu Tests naimplementována funkce Soap, která umožní rychlejší komunikaci s aplikacemi v kontroléru. Tato vlastnost umožní rychlé roztáčení motoru během testování restartů v Man konfiguraci bez použití interních funkcí konfiguračního nástroje Scout. Otevřenou otázkou pro zvýšení produktivity je i vhodnější rozpracování dávkového spouštění automatických testů, vhodnou cestou by mohlo být spouštění testů ve více vláknech a jejich kontrola hlavním programem.

Z důvodu ochrany vlastnictví společnosti ANF DATA spol. s.r.o., a Siemens Copany nemohou být součástí této diplomové práce realizované projekty v plné šíři (zdrojové kódy, exporty projektů).

Použitá literatura

[1] Kosek, R.: PROFINET – *standard pro průmyslový Ethernet v automatizaci*, časopis Automa, 2005/4, dostupné z internetových stránek http://www.odbornecasopisy.cz/index.php?id_document=30419

[2] Vopička, J.: *PROFINET IRT v problematice řízení pohybu a konfigurace*, interní prezentace společnosti ANF DATA spol. s.r.o., a Siemens Company

[3] Čermák, J.: *PROFINET Basic*, interní prezentace společnosti ANF DATA spol. s.r.o., a Siemens Company

[4] Weber, K.; Popp, M,: *The Rapid Way to PROFINET*, PROFIBUS Nutzerorganisation e V. 2004

[5] Prezentace: *Simotion, Stručný přehled*, interní prezentace společnosti ANF DATA spol. s.r.o., a Siemens Company

Seznam příloh

I. Fotka MAN racku **1**

II. Fotka sestavy pro projekt 5xP350 a P350 + 2xET200s **2**

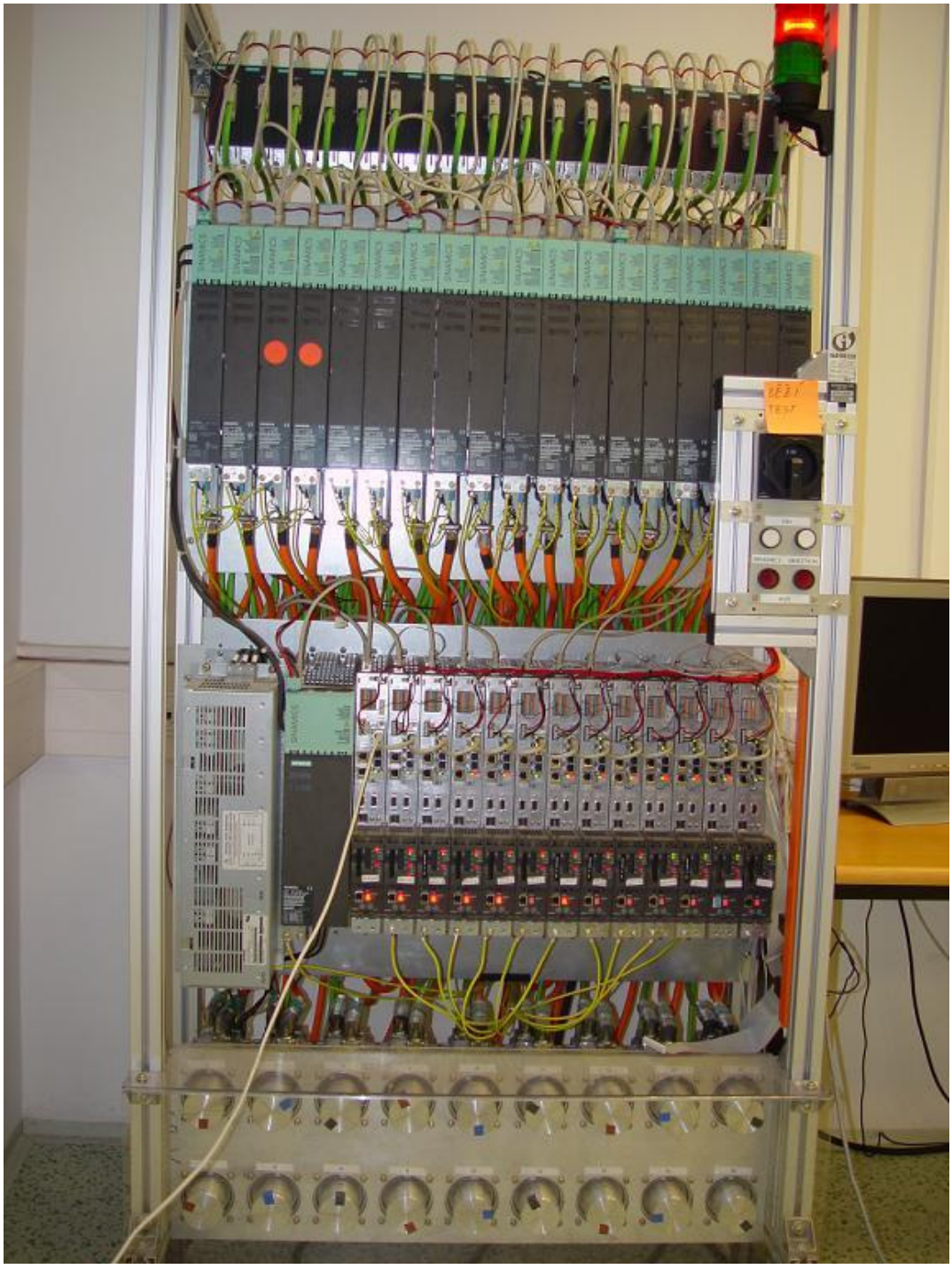
III. Screenshoot obrazovky z průběhu automatického testu pro změnu komunikačního cyklu **3**

IV. Ukázka z konfiguračního prostředí Scout **4**

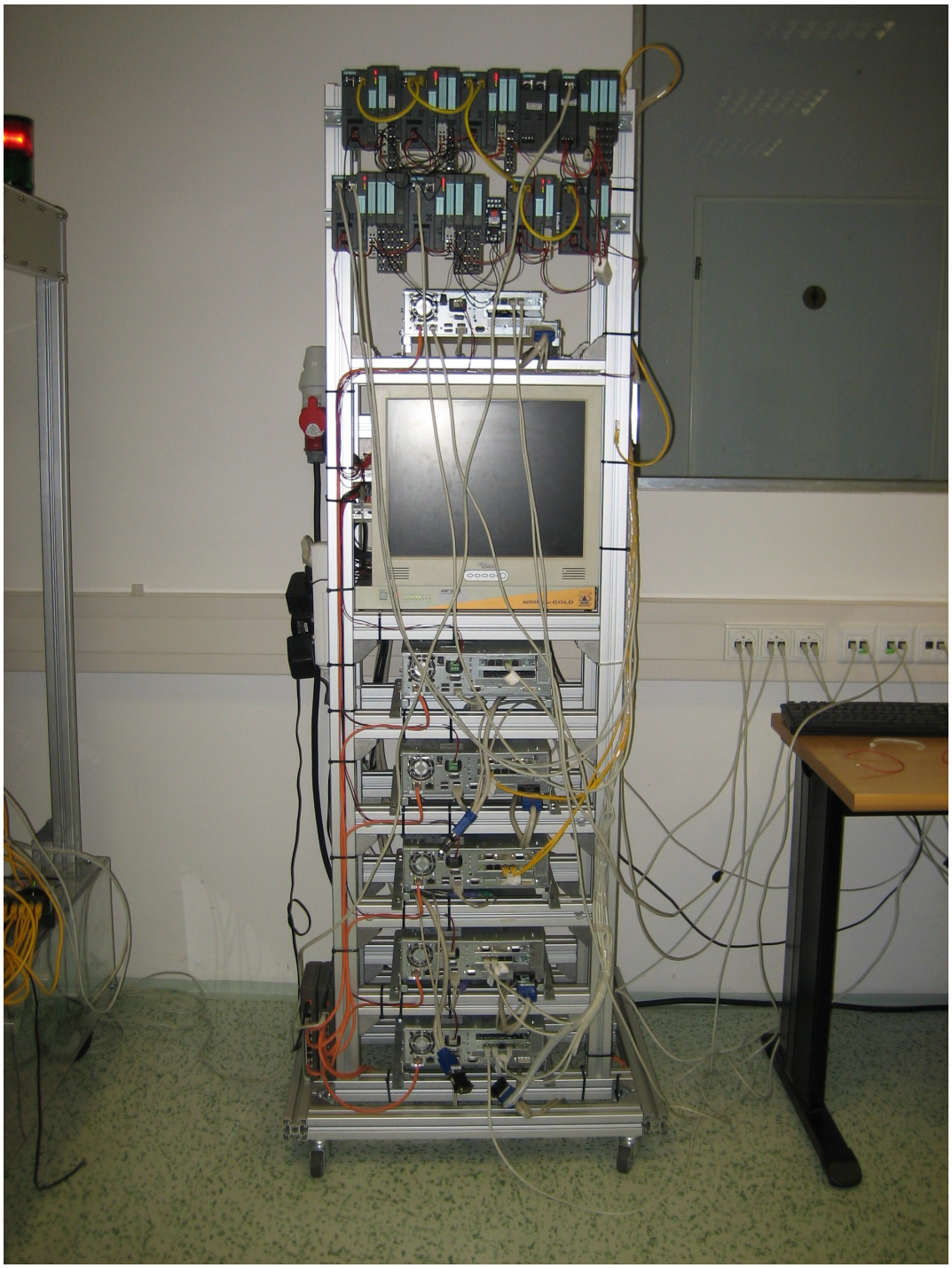
V. CD, které obsahuje:

- Elektronickou formu této Diplomové práce
- Logovací soubory automatických testů

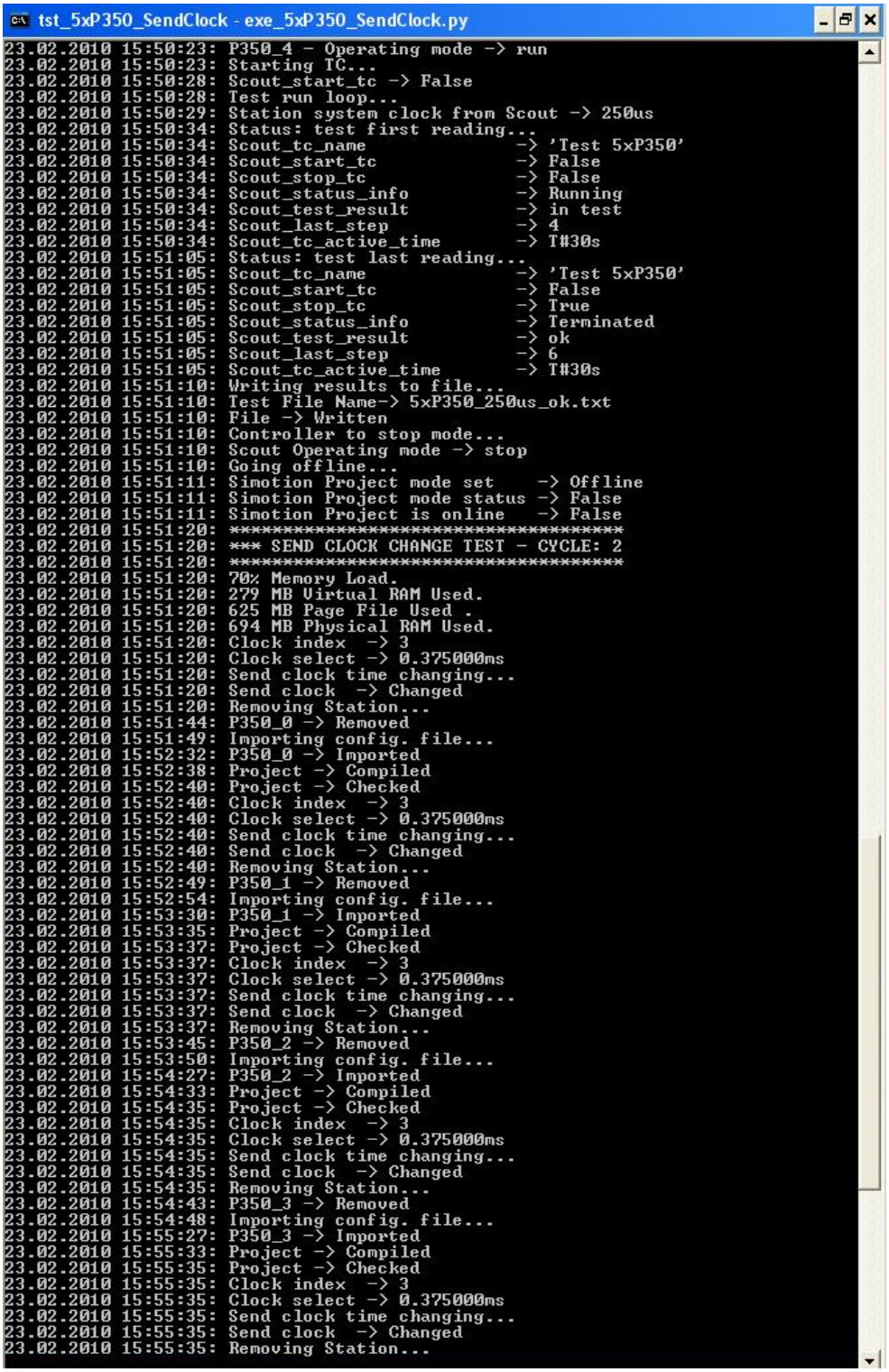
I. Fotka MAN racku



II. Fotka sestavy pro projekt 5xP350 a P350 + 2xET200s



III. Screenshoot z průběhu automatického testu pro změnu komunikačního cyklu



IV. Ukázka z konfiguračního prostředí Scout